MULTIDIMENSIONAL CONVOLUTIONAL CODES

A Dissertation

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Paul A. Weiner, B.A., M.A.

Joachim Rosenthal, Director

Department of Mathematics

Notre Dame, Indiana

April 1998

Acknowledgements

I would like to thank many people who helped and encouraged me with this work. My advisor, Joachim Rosenthal, worked with me long and patiently so that I could produce this dissertation. My committee members, Heide Glüsing-Lüerßen, Alan Howard, and Juan Migliore, made many suggestions that have improved this work greatly. My academic "siblings", Brian Allen, Roxana Smarandache, and Eric York helped me in many ways, from technical assistance with my computer to explanations of some of the finer points of coding theory, but most importantly with their friendship. The faculty, staff, and graduate student body of the Notre Dame Department of Mathematics made working toward my Ph.D. a great experience.

Chris Peterson gave me multiple explanations of various topics in algebraic geometry and commutative algebra. M. S. Ravi pointed me in the right direction regarding some differences between 2-dimensional and higher-dimensional codes. Conversations with Philippe Loustaunau got me started thinking about ways of calculating the distance of a multidimensional convolutional code.

Saint Mary's University of Minnesota gave me both financial support and sufficient time to pursue graduate studies. I received financial support from the University of Notre Dame through the Mathematics Department and through a fellowship from the Center for Applied Mathematics. I also received funding from the NSF through my advisor's grant (DMS-96-10389).

Finally without the love, support, and patience of my family this project would have been totally beyond my grasp.

Abstract

In this dissertation some of the theory of multidimensional convolutional codes is developed.

Given a finite field, $\mathbb{F}$, consider the polynomial ring $R = \mathbb{F}[z_1, ..., z_m]$ in m indeterminates over $\mathbb{F}$. An m-dimensional convolutional code of length n is an R-submodule of R^n. With this definition, many results and techniques from commutative algebra and algebraic geometry are available for the study of m-dimensional convolutional codes. These connections are explored in this dissertation (especially in Chapter 3).

Much attention in the literature has been given to 1-dimensional convolutional codes, a slight amount to 2-dimensional convolutional codes, and almost none to higher-dimensional convolutional codes. It is worth noting that m-dimensional convolutional codes are a nontrivial generalization of the 1-dimensional case. There are interesting differences between the 1- and 2-dimensional cases, and again between the 2- and 3-dimensional cases. Many of these differences are considered in this work.

A code construction considered in Chapter 4 gives a family of codes for which a lower distance bound is available. The lower bound is derived using the idea of a monomial ordering, and hence is related to Gröbner basis theory. A decoding algorithm for this type of code is also given.

Dedication

To Kathy, Ben, and Laura
with love and thanks.
And in memory of
Nella and Milton.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 A Brief Sketch of Coding Theory

The aim of coding theory is to analyze, design, and implement methods of communicating over an unreliable channel. The point being that when information is transmitted errors may occur. Consider the children's game of "telephone" where the first child whispers a message to the second child who whispers what she heard to the third child, and so on down the line. Finally the last child says what he thinks the message is. The results are often amusing. I remember playing this game with the initial message being "Please pour me some milk", but the final message was "Please poke me in the eye." A chain of whispers is an unreliable communication channel.

This game has serious overtones, for precise communication is important in many areas of life. We expect clean telephone links; we want clear television pictures; we demand high quality sound on compact disc recordings. But in real life errors occur, and in many instances coding theory can help to detect and correct errors in the transmission of information.

When there is an error in communication, one of three things may happen: the receiver may not realize that an error has occurred (an undetected error), or the receiver may know that an error has occurred but not know what the error is (a detected but uncorrected error), or the receiver may be able to determine what the sender intended to transmit in spite of the error (a corrected error). The English language serves as a communication system, and of course errors may occur. If Alice wanted to tell Bob that her cat had caught a mouse, she might try to e-mail him, "My cat caught a mouse." If for some reason Bob received the message, "My cat caught a cold," Bob would think this a perfectly reasonable message, and the error would go undetected. If Bob received, "My cat caught a qwertyuiop," he would know that an error had occurred, but he would not know what the error was. The error would be detected but not corrected. Finally if Bob received the message, "My cat caught a mousn," he would guess that he was supposed to have received, "My cat caught a mouse." The error would have been corrected.

What makes error detection and correction possible in a communication system is redundancy. We can often correct a misspelled word because the english language is redundant. Of course redundancy carries a price, namely efficiency. One of the main goals of coding theory is to design communication systems so that a certain level of protection from errors is achieved while introducing the least possible amount of redundancy. For instance seven binary digits can serve to protect four binary digits from one error (so three redundant bits are used).

The fields of information theory and coding theory trace their roots to the 1948 paper of Claude Shannon [45]. A general mathematical framework for encoding and decoding is as follows.

A set $\mathcal{M}$ of message words and a set $\mathcal{Y}$ of receivable words are given along with an injective set function

$$\varphi : \mathcal{M} \longrightarrow \mathcal{Y}.$$

The set $\mathcal{C} = \varphi(\mathcal{M}) \subseteq \mathcal{Y}$ is the code. Elements of $\mathcal{C}$ are called codewords. The function φ is an encoder.

For decoding, a set $\mathcal{D}$ with $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{Y}$ is specified, along with a function

$$\psi : \mathcal{D} \longrightarrow \mathcal{M} \quad \text{such that} \quad \psi|_{\mathcal{C}} = \varphi^{-1}.$$

ψ is a decoder corresponding to φ.

A message word $v \in \mathcal{M}$ is encoded to $w = \varphi(v) \in \mathcal{C}$. The codeword w is then transmitted over some transmission channel, and a word $y \in \mathcal{Y}$ is received. Due to noise in the transmission channel, y need not equal w.

If y is an element of $\mathcal{D}$, then the receiver computes $\tilde{v} = \psi(y)$. If $\tilde{v} = v$, then the original message is recovered. We note that this is automatically the case if $y = w$ (for then $y = w = \varphi(v) \in \mathcal{C}$, and $\psi|_{\mathcal{C}} = \varphi^{-1}$).

Two things may go wrong in the decoding process: If $y \notin \mathcal{D}$, then decoding cannot even be attempted (for instance, if you read the supposed English word 'gxrqe', you probably would have no idea what word was actually intended). The second thing that may go wrong is that even if $y \in \mathcal{D}$, it is possible that $\psi(y) \neq v$, causing the wrong message word to be recovered. (For instance, if you read the word 'stip', you might guess that 'stop' was intended; however, if in fact 'skip' was intended, an error in decoding would have occurred.)

If $\mathcal{Y}$ is an abelian group (denoted additively), and if w is transmitted and y is received, then an error $e = y - w \in \mathcal{Y}$ has occurred (of course if $w = y$, then $e = 0$, signifying an error-free transmission).

Two of the most important types of codes studied and used today are block codes and convolutional codes. A block code consists of a fixed finite alphabet (often the binary alphabet, $\{0, 1\}$) and a set of strings of fixed length, n, of symbols from the alphabet. These strings are called codewords. For example, $\mathcal{C} = \{0110, 1010, 1101\}$ is a block code of length 4 over the binary alphabet. This code contains three codewords. (In this example we are suppressing the space of message words, $\mathcal{M}$, but presumably it would consist of three elements. Also we are suppressing the space $\mathcal{Y}$ of receivable words, but it might be the set of all strings of four binary digits.) If we received as a message the block 1100, we would recognize that one or more errors had occurred in transmission as 1100 is not a codeword. We might correct the error to 1101 because that correction involves changing only one bit (whereas correcting 1100 to 0110 or to 1010 requires changing two bits). We generally assume that the correct codeword is the one that involves the fewest bit changes. The most important class of block codes is the class of *linear block codes*. A linear block code is a vector subspace of ${\mathbb{F}_q}^n$ where $\mathbb{F}_q$ is the finite field with q elements. The area of block codes has been well studied, though the search for new classes of block codes with good error detection and correction properties and good decoding schemes continues. Hill [17] gives an excellent introduction to block codes. More advanced texts include those by MacWilliams and Sloane [28], and Roman [42].

A convolutional code may be regarded as an $\mathbb{F}_q[z]$-submodule of $\mathbb{F}_q[z]^n$, where $\mathbb{F}_q[z]$ is a polynomial ring in one indeterminate over $\mathbb{F}_q$. This definition follows the development by York in [55]. In convolutional coding, information is not transmitted in fixed-length blocks, but rather in a stream of data. Foundational work in the theory of convolutional codes was done by Forney [12, 13, 14]. The search continues for convolutional codes with good encoding and decoding properties. The book by Lin and Costello [26] is a good introduction to both block codes and convolutional codes. The book [35] by Piret gives a good treatment of convolutional codes.

Linear block codes and convolutional codes are examples of *linear* codes. A general framework for studying linear codes is to regard them as R-submodules of R^n where R is a commutative noetherian ring. (Then linear block codes are linear codes using the ring $R = \mathbb{F}_q$.) One benefit to adopting this point of view is that for many such rings there is then a duality between codes and systems. This duality is treated in detail by Oberst [33]. We will sketch this duality in Section 2.6. Several recent papers have looked at various codes from this module-theoretic point of view with different rings R.

In [8, 47] Fornasini and Valcher look at "2-dimensional convolutional codes" using the Laurent polynomial ring $R = \mathbb{F}_q[z_1, z_2, {z_1}^{-1}, {z_2}^{-1}]$. The papers [9, 10, 11] by the same authors are also relevant as they consider "finite support nD behaviors", which though considered from a somewhat different point of view are, in the terminology we are employing here, convolutional codes using the Laurent polynomial ring in n indeterminates. In her Ph.D. dissertation [48], Walker studies linear codes where R is an Artinian local ring. In fact, before specializing to this case, Walker suggests considering letting a code be any R-submodule of R^n where R is *any* ring. As mentioned above, York [55] studies codes over the polynomial ring $\mathbb{F}_q[z]$ in one variable. Oberst [33] considers systems (which are related to codes as discussed by York [55] and below) over the ring $R = \mathbb{F}[z_1, ..., z_m]$ where $\mathbb{F}$ is any field.

Multidimensional convolutional codes generalize convolutional codes to polynomial rings in several variables.

These codes may prove useful in transmission of m-dimensional data (especially in the 2-D case, where for instance photographs and video images may be transmitted as 2-D data). No doubt higher dimensional codes will find further applications in the future.

Some work, specialized to the 2-D case, has been done by Fornasini and Valcher in [8, 47]. Interestingly, there are significant theoretical differences between 1- and 2-D convolutional codes. And surprisingly, there are also important differences between 2- and 3-D convolutional codes. These differences will be explored below.

It is my aim in writing this dissertation to develop the theory of m-D convolutional codes, especially highlighting the similarities and differences with 1-D convolutional codes.

1.2 Outline of Dissertation

In Chapter 2 we introduce the basic notions of multidimensional convolutional codes, explicitly showing them as finitely generated modules and as finite sequence spaces with shift and linearity properties. We make the observation that any multidimensional convolutional code may be represented as the row space of a polynomial matrix and conversely. This material is standard, though largely unexplored for dimension higher than 2. Also we define weight and distance for a multidimensional convolutional code, again a standard notion, but not much studied for higher dimensions. We derive a new upper bound on the distance of an m-D convolutional code based on upper bounds of block codes. Finally in the last section of the chapter we review an important duality between multidimensional behavioral systems and multidimensional convolutional codes.

In Chapter 3 we take a detailed look at the matrix theory and module theory that forms much of the algebraic material that is foundational to our subject. Many differences between 1-D and 2-D codes and between 2-D and higher-D codes are explored in this chapter. In the first section we give definitions and basic results for three notions of primeness for a polynomial matrix in several indeterminates. In the second section of this chapter we take an excursion into the theory and practice of determinantal matrix factorization. New results here include the fact that if a ring has a determinantal factorization property for square matrices, then it has that factorization property for arbitrary matrices. We also give a new matrix factorization technique. And we reinterpret an older result in the literature that allows us to distinguish between a polynomial ring in one variable over a PID and a polynomial ring in one variable over a UFD. In the third section of Chapter 3 we consider conditions under which a convolutional code has a kernel representation. This result also appears in the very recent papers [10, 54]. We believe that our proof is new, and also it is somewhat more general than those in the references given. In the fourth section of this chapter we apply the earlier results to multidimensional convolutional codes, deriving some interesting differences among codes of different dimensions. In the final section of Chapter 3 we consider some further results on matrix primeness and matrix factorization.

In Chapter 4 we generalize ideas that have been lightly studied in the case of 1-D codes, and use them to define what we call unit memory codes. In the case where the unit memory code is constructed from block codes, we give a lower bound on the distance in terms of the distances of the component block codes. In this case we also give a decoding algorithm based on being able to decode the constituent block codes. The distance bound and decoding scheme discussed in this chapter are new, even in the 1-D case.

In the 1-dimensional coding literature the notion of a catastrophic code is studied. A code is catastrophic if it is possible for finitely many errors in transmission to cause infinitely many errors in decoding. In Chapter 5 we give a necessary and sufficient condition for a multidimensional convolutional code to be catastrophic.

In Chapter 6 we give a brief summary of this work. Also we give several topics and questions about m-D convolutional codes that merit further consideration.

CHAPTER 2

MULTIDIMENSIONAL CODES

2.1 Notation and Definitions

We begin by setting some notation. Let $\mathbb{N}$ be the set of nonnegative integers. Let $\mathbb{F} = \mathbb{F}_q$ be the finite field with q elements. Let $R = \mathbb{F}[z_1, ..., z_m]$ be the polynomial ring in m indeterminates over $\mathbb{F}$. Let R^n denote the space of n-component row vectors with entries in R. Let $Q(R)$ be the field of fractions of R. Let

$$\mathcal{S} = \{w : \mathbb{N}^m \longrightarrow \mathbb{F} \mid w \text{ has finite support}\}.$$

(To say $w : \mathbb{N}^m \longrightarrow \mathbb{F}$ has finite support means that $w(i_1, ..., i_m) = 0$ for all but finitely many $(i_1, ..., i_m) \in \mathbb{N}^m$). $\mathcal{S}$ is the *m-dimensional finite sequence space of* $\mathbb{F}$.

There is an $\mathbb{F}$-isomorphism

$$\begin{array}{rccl} \psi : & \mathcal{S} & \longrightarrow & R \\ & w & \longmapsto & \displaystyle\sum_{(i_1,...,i_m)} w(i_1, ..., i_m) \cdot {z_1}^{i_1} \cdot ... \cdot {z_m}^{i_m}. \end{array}$$

Here we associate points of the integer lattice $\mathbb{N}^m$ with monomials in R via the correspondence

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

An element of $\mathbb{F}$ attached to a point of the lattice becomes a coefficient of the monomial.

ψ induces an $\mathbb{F}$-isomorphism

$$\begin{array}{rccl} \psi^n : & \mathcal{S}^n & \longrightarrow & R^n \\ & (w_1, ..., w_n) & \longmapsto & (\psi(w_1), ..., \psi(w_n)). \end{array}$$

Now for $w \in \mathcal{S}$ we write $z_k w$ for $\psi^{-1}(z_k \psi(w))$. I.e., $z_k : \mathcal{S} \longrightarrow \mathcal{S}$ is the k^{th} right shift operator on $\mathcal{S}$.

Example 2.1.1 . Over $\mathbb{F}_3$

$$\begin{array}{cccc} \bullet & \bullet_2 & \bullet & \bullet \\ \bullet_1 & \bullet & \bullet_2 & \bullet \end{array} \quad \xrightarrow{\psi} \quad 1 + 2{z_1}^2 + 2z_1 \cdot z_2$$

$$\downarrow z_1 \cdot$$

$$\begin{array}{cccc} \bullet & \bullet & \bullet_2 & \bullet \\ \bullet & \bullet_1 & \bullet & \bullet_2 \end{array} \quad \xleftarrow{\psi^{-1}} \quad z_1 + 2{z_1}^3 + 2{z_1}^2 \cdot z_2 \ .$$

Definition 2.1.2 An *m-dimensional (m-D) code of length n over* $\mathbb{F}$ is a subset $C \subseteq \mathcal{S}^n$. An element $w \in C$ is a *codeword.*

Remark 2.1.3 We may consider a code C as being a subset of R^n via the isomorphism ψ^n. Then codewords are n-vectors of polynomials.

Definition 2.1.4 A code C is *linear* if it is an $\mathbb{F}$-linear subset of R^n. C is *right shift invariant* if $z_k \cdot C \subseteq C$ for $k = 1, ..., m$.

Proposition 2.1.5 *A code $C \subseteq \mathcal{S}^n$ is linear and right shift invariant if and only if $\psi^n(C)$ is an R-submodule of R^n.*

Proof: Because ψ^n is an $\mathbb{F}$-isomorphism, it follows that C is linear if and only if $\psi^n(C)$ is an $\mathbb{F}$-linear subset of R^n. Also right shift invariance of C is equivalent to closure of $\psi^n(C)$ under multiplication by z_i $(i = 1, ..., m)$. However, this along with $\mathbb{F}$-linearity of $\psi^n(C)$ is equivalent to $\psi^n(C)$ being an R-submodule of R^n. $\square$

Definition 2.1.6 An *m-D convolutional code of length n over* $\mathbb{F}$ is a linear, right shift invariant code of length n over $\mathbb{F}$. Equivalently, an m-D convolutional code of length n is an R-submodule of R^n.

From now on we will let $\mathcal{C}$ be an m-D convolutional code.

Remark 2.1.7

i) A 0-D convolutional code is a linear block code.

ii) A 1-D convolutional code is a 'convolutional code' as described in the introduction.

iii) So the present work is a generalization of what currently appears in the coding theory literature.

2.2 Generator Matrices and Encoders

Any m-D convolutional code, $\mathcal{C}$, of length n is a finitely generated R-module. So there is a positive integer l and a polynomial matrix $G \in R^{l \times n}$ such that $\mathcal{C} = \text{rowspace}\,(G) = R^l \cdot G$. Such a matrix G is a *generator matrix* of $\mathcal{C}$.

Definition 2.2.1 Given a polynomial matrix $G \in R^{l \times n}$, let $\mathcal{C}(G) := \text{rowspace}\,(G)$. The *rank* of $\mathcal{C} = \mathcal{C}(G)$ is the largest integer k for which there is a nonzero $k \times k$ minor of G. We write $\text{rank}(\mathcal{C}) = k$.

Equivalently, the rank of $\mathcal{C}$ is the dimension of the vector subspace generated by the rows of G in $Q(R)^n$ over the field $Q(R)$.

If $\text{rank}(\mathcal{C}) = k$, then the *rate* of $\mathcal{C}$ is $\frac{k}{n}$.

By virtue of the equivalence in Definition 2.2.1, it follows that the rank of $\mathcal{C}$ is independent of the generator matrix chosen for $\mathcal{C}$.

Definition 2.2.2 The m-D convolutional code $\mathcal{C}$ is *free* if it is free as an R-module.

If $\mathcal{C}$ is free of rate $\frac{k}{n}$, then $\mathcal{C} = \mathcal{C}(G)$ for some $G \in R^{k \times n}$. In this case we say G is an *encoder* of $\mathcal{C}$.

Remark 2.2.3 If $m = 1$, then R is a principal ideal domain (PID), and hence any submodule of R^n is free (see for instance [18]). So any 1-D convolutional code is free.

Proposition 2.2.4 *Suppose $\mathcal{C}$ is a free convolutional code of rate $\frac{k}{n}$ with encoder $G \in R^{k \times n}$. Then*

$$0 \longrightarrow R^k \xrightarrow{\cdot G} R^n \xrightarrow{\pi} R^n/\mathcal{C} \longrightarrow 0$$

is a short exact sequence.

Proof: Because $\mathcal{C}$ is free of rank k and $G \in R^{k \times n}$, it follows that any nonzero linear combination over R of rows of G is nonzero. Therefore the left kernel of G consists of the zero vector only. So the map $R^k \xrightarrow{\cdot G} R^n$ is injective. By definition, the image of this map is $\mathcal{C}$. Exactness of the sequence now follows immediately from standard properties of the projection map π. $\square$

Definition 2.2.5 A square matrix $U \in R^{k \times k}$ is *unimodular* if U is a unit in $R^{k \times k}$ considered as a matrix ring. That is U is unimodular if and only if there is a matrix $V \in R^{k \times k}$ such that $U \cdot V = V \cdot U = I_k$. Also U is unimodular if and only if $\det(U)$ is a unit in R (i.e., a nonzero element of $\mathbb{F}$).

Let $G, G_1 \in R^{k \times n}$ both be of full row rank. We say G and G_1 are *equivalent encoders if* $\mathcal{C}(G) = \mathcal{C}(G_1)$.

Proposition 2.2.6 *Let $G \in R^{l \times n}$, $G_1 \in R^{l_1 \times n}$ (not necessarily of full row rank). $\mathcal{C}(G_1) \subseteq \mathcal{C}(G)$ if and only if there exists a matrix $T \in R^{l_1 \times l}$ such that $T \cdot G = G_1$.*

Proof: Simply observe that $\mathcal{C}(G_1) \subseteq \mathcal{C}(G)$ if and only if rowspace $(G_1) \subseteq$ rowspace (G). This in turn occurs if and only if each row of G_1 can be written as an R-combination of rows of G. This last is equivalent to the statement of the proposition. $\square$

The following corollary is an immediate consequence.

Corollary 2.2.7 *Let $G, G_1 \in R^{k \times n}$ be of full row rank. G and G_1 are equivalent encoders if and only if there exists a unimodular matrix $U \in R^{k \times k}$ such that $U \cdot G = G_1$.*

2.3 Examples

In this section we give several examples to clarify the above ideas. All examples will use the binary ground field $\mathbb{F} = \mathbb{F}_2$. All examples will use $R = \mathbb{F}[z_1, z_2]$ and hence be 2-dimensional.

Example 2.3.1 $G(z_1, z_2) = \begin{bmatrix} 1 + z_1 z_2 + {z_1}^3 & z_1 + {z_2}^2 \end{bmatrix}$.
$\mathcal{C}(G)$ is a free rate $\frac{1}{2}$ code with encoder G.

Example 2.3.2 Let $G(z_1, z_2) = \begin{bmatrix} z_1 z_2 & z_1 + {z_1}^2 z_2 & {z_1}^2 \\ z_2 + z_1 z_2 & 1 + z_1 + z_1 z_2 + {z_1}^2 z_2 & z_1 + {z_1}^2 \end{bmatrix}$.

$\mathcal{C}(G)$ is free of rate $\frac{1}{3}$. An encoder is $G_1 = \begin{bmatrix} z_2 & 1 + z_1 z_2 & z_1 \end{bmatrix}$. Observe that $G = \begin{bmatrix} z_1 \\ 1 + z_1 \end{bmatrix} \cdot G_1$, and $G_1 = \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot G$; so by Proposition 2.2.6, G and G_1 generate the same code.

Example 2.3.3 Let $G(z_1, z_2) = \begin{bmatrix} {z_1}^2 & z_1 & z_1 z_2 \\ z_1 + z_1 z_2 & 1 + z_2 & z_2 + {z_2}^2 \end{bmatrix}$.

It can easily be shown that $\mathcal{C}(G)$ is rate $\frac{1}{3}$ but not free. That is, $\mathcal{C}(G)$ has no 1×3 encoder.

$$\mathcal{C}\left(\begin{bmatrix} z_1 & 1 & z_2 \end{bmatrix}\right)$$

is a free rate $\frac{1}{3}$ code that properly contains $\mathcal{C}(G)$.

Example 2.3.4 In this example we look at the encoding process. Let

$$G = \begin{bmatrix} 1 & z_1 & z_2 & 0 \\ 0 & 1 & z_1 & 1 \\ 0 & 0 & 1 & z_2 \end{bmatrix}.$$

Rank(G)=3, and so $\mathcal{C}(G)$ is free of rate $\frac{3}{4}$. Also this code is two-dimensional. So on the sequence level we can encode an assignment of binary triples to points of $\mathbb{N}^2$ where all but finitely many of the triples are 000. The coded output will consist of an assignment of binary quadruples to points of $\mathbb{N}^2$ where all but finitely many of the quadruples are 0000. Because of linearity and shift invariance, encoding is completely determined by what happens to 100 assigned to $(0,0)$ and 000 everywhere else, and to 010 assigned to $(0,0)$ and 000 everywhere else, and to 001 assigned to $(0,0)$ and 000 everywhere else. For these three sequences we have the following encoding (in all diagrams the lower left point is $(0,0) \in \mathbb{N}^2$ and points not shown are assigned all zeros).

$$\begin{array}{ccc} \bullet_{000} & \bullet_{000} & \bullet_{000} \\ \bullet_{000} & \bullet_{000} & \bullet_{000} \\ \bullet_{100} & \bullet_{000} & \bullet_{000} \end{array} \quad \xmapsto{\cdot G} \quad \begin{array}{ccc} \bullet_{0000} & \bullet_{0000} & \bullet_{0000} \\ \bullet_{0010} & \bullet_{0000} & \bullet_{0000} \\ \bullet_{1000} & \bullet_{0100} & \bullet_{0000} \end{array} ,$$

$$\begin{array}{ccc} \bullet_{000} & \bullet_{000} & \bullet_{000} \\ \bullet_{000} & \bullet_{000} & \bullet_{000} \\ \bullet_{010} & \bullet_{000} & \bullet_{000} \end{array} \quad \xmapsto{\cdot G} \quad \begin{array}{ccc} \bullet_{0000} & \bullet_{0000} & \bullet_{0000} \\ \bullet_{0000} & \bullet_{0000} & \bullet_{0000} \\ \bullet_{0101} & \bullet_{0010} & \bullet_{0000} \end{array} ,$$

$$\begin{array}{ccc} \bullet_{000} & \bullet_{000} & \bullet_{000} \\ \bullet_{000} & \bullet_{000} & \bullet_{000} \\ \bullet_{001} & \bullet_{000} & \bullet_{000} \end{array} \quad \xmapsto{\cdot G} \quad \begin{array}{ccc} \bullet_{0000} & \bullet_{0000} & \bullet_{0000} \\ \bullet_{0001} & \bullet_{0000} & \bullet_{0000} \\ \bullet_{0010} & \bullet_{0000} & \bullet_{0000} \end{array} .$$

From a polynomial point of view, these maps respectively correspond to

$$\begin{array}{ccc} [1,0,0] & \xmapsto{\cdot G} & [1, z_1, z_2, 0] \\ [0,1,0] & \xmapsto{\cdot G} & [0, 1, z_1, 1] \\ [0,0,1] & \xmapsto{\cdot G} & [0, 0, 1, z_2] \ . \end{array}$$

As noted above, the action of $\cdot G$ is now completely determined. For instance,

$$\begin{array}{ccc} \bullet_{000} & \bullet_{000} & \bullet_{000} \\ \bullet_{101} & \bullet_{001} & \bullet_{000} \\ \bullet_{011} & \bullet_{110} & \bullet_{010} \end{array} \quad \xmapsto{\cdot G} \quad \begin{array}{cccc} \bullet_{0011} & \bullet_{0001} & \bullet_{0000} & \bullet_{0000} \\ \bullet_{1011} & \bullet_{0100} & \bullet_{0000} & \bullet_{0000} \\ \bullet_{0111} & \bullet_{1111} & \bullet_{0011} & \bullet_{0010} \end{array} .$$

From the polynomial point of view, this corresponds to

$$\left[\begin{array}{ccc} z_1 + z_2 & 1 + z_1 + z_1^2 & 1 + z_2 + z_1 z_2 \end{array} \right] \quad \xmapsto{\cdot G}$$

$$\left[\begin{array}{cccc} z_1 + z_2 & 1 + z_1 + z_1 z_2 & 1 + z_1 + z_1^2 + z_1^3 + z_2 + z_2^2 & 1 + z_1 + z_1^2 + z_2 + z_2^2 + z_1 z_2^2 \end{array} \right] .$$

2.4 Weight and Distance

In this section we give definitions of weight and distance. These generalize the definition of Hamming weight and distance for a block code or for a 1-D convolutional code (see for instance Lin and Costello [26]).

We briefly review multi-index notation. Let $\alpha = (\alpha_1, ..., \alpha_m) \in \mathbb{N}^m$. By $\mathbf{z}^\alpha$ we mean $z_1^{\alpha_1} \cdot \ldots \cdot z_m^{\alpha_m}$. Then a polynomial $f \in R$ may be written as

$$f(z_1, ..., z_m) = \sum_{\alpha \in \mathbb{N}^m} f_\alpha z^\alpha \text{ where } f_\alpha \in \mathbb{F} \ .$$

Similarly a vector $w = \left[\begin{array}{ccc} w_1 & \ldots & w_n \end{array} \right] \in R^n$ may be written as

$$w = \sum_{\alpha \in \mathbb{N}^m} w_\alpha z^\alpha \text{ where } w_\alpha \in \mathbb{F}^n .$$

Example 2.4.1 If $w = \left[\begin{array}{ccc} 1 + z_1^2 & 1 + z_1 z_2 + z_1^2 & z_2 + z_1 z_2 \end{array} \right]$, then

$$w = \left[\begin{array}{ccc} 1 & 1 & 0 \end{array} \right] + \left[\begin{array}{ccc} 0 & 0 & 1 \end{array} \right] z_2 + \left[\begin{array}{ccc} 1 & 1 & 0 \end{array} \right] z_1^2 + \left[\begin{array}{ccc} 0 & 1 & 1 \end{array} \right] z_1 z_2 \ .$$

The most important measure of weight and distance for a code is the Hamming weight and distance. We give the definitions as they apply in the m-D case.

Definition 2.4.2 Let $a \in \mathbb{F}^n$. The *weight* of a is given by the number of nonzero entries of a. It is denoted by $\mathrm{wt}(a)$.

Let $f \in R$. The *weight* of f is the number of nonzero terms of f. It is denoted $\mathrm{wt}(f)$.

Let $w = \left[\begin{array}{ccc} w_1 & \dots & w_n \end{array}\right] \in R^n$. Then the *weight* of w is given by

$$\operatorname{wt}(w) = \sum_{j=1}^{n} \operatorname{wt}(w_j)\,.$$

Equivalently, if

$$w = \sum_{\alpha \in \mathbb{N}^m} b_\alpha z^\alpha \text{ where } b_\alpha \in \mathbb{F}^n\,,$$

then

$$\operatorname{wt}(w) = \sum_{\alpha \in \mathbb{N}^m} \operatorname{wt}(b_\alpha)\,.$$

So in Example 2.4.1 above, $wt(w) = 7$, and this can be obtained either by counting nonzero terms in the first representation of w or by counting nonzero vector entries in the second.

Definition 2.4.3 Given two elements $w, \tilde{w} \in R^n$ the *(Hamming) distance* between w and $\tilde{w}$ is given by $\operatorname{dist}(w, \tilde{w}) = \operatorname{wt}(w - \tilde{w})$. Given any m-D code (not necessarily convolutional) C of length n, the *distance of* C is defined as

$$\operatorname{dist}(C) = \min\{\operatorname{dist}(w, \tilde{w}) : w, \tilde{w} \in C, w \neq \tilde{w}\}.$$

Remark 2.4.4

i) The Hamming distance gives a metric on R^n called the Hamming metric.

ii) For a convolutional code $\mathcal{C}$ we have that $\operatorname{dist}(\mathcal{C}) = \min\{\operatorname{wt}(w) : w \in \mathcal{C}, w \neq 0\}$. This is because $\operatorname{dist}(w_1, w_2) = \operatorname{wt}(w_1 - w_2)$, and $w_1 - w_2 \in \mathcal{C}$ whenever $w_1, w_2 \in \mathcal{C}$.

The following result is standard in coding theory. It follows immediately from the definition of Hamming distance and from the properties of a metric space.

Lemma 2.4.5 *Let $\mathcal{C}$ be a convolutional code with $d = \operatorname{dist}(\mathcal{C})$. Let $t = \lfloor \frac{d-1}{2} \rfloor$ where $\lfloor x \rfloor$ denotes the greatest integer that is less than or equal to x. Let $y \in R^n$. If $w \in \mathcal{C}$ is a codeword such that $\operatorname{dist}(w, y) \leq t$, then w is the unique codeword nearest (with respect to the Hamming metric) to y.*

We say that $\mathcal{C}$ can correct up to t errors. (We may or may not have an efficient algorithm for actually correcting the errors. If we do, we say we can decode up to t errors.)

2.5 The Block Code Bound

In this section we derive an upper bound on the distance of an m-D convolutional code. This generalizes to the m-D case a 1-D result that may be found in the paper [25] by Lee.

Proposition 2.5.1 *Let $G \in R^{k \times n}$. Let $\mathcal{C} = \mathcal{C}(G)$ be the m-D convolutional code generated by G. Write*

$$G = \sum_{i=1}^{r} G_i \mathbf{z}^{\alpha_i}\,,$$

where $G_i \in \mathbb{F}^{k \times n}$, and $\alpha_i \in \mathbb{N}^m$ is a multi-index with $\alpha_i \neq \alpha_j$ for $i \neq j$ $(1 \leq i, j \leq r)$. (That is, G may be written with only r different monomials among its entries.)

Let $\hat{G} = [G_1 \mid G_2 \mid \dots \mid G_r] \in \mathbb{F}^{k \times (rn)}$. Let C be the linear block code generated by $\hat{G}$ (i.e., C is the subspace of $\mathbb{F}^{rn}$ spanned by the rows of $\hat{G}$). We allow the possibility that $\dim(C) < k$, but we observe that in any event, $C = \mathbb{F}^k \hat{G}$.

Then $\operatorname{dist}(\mathcal{C}) \leq \operatorname{dist}(C)$.

Proof: Suppose C has a word of weight p. That is, there exists $v \in \mathbb{F}^k$ such that $\mathrm{wt}(v\hat{G}) = p$. We note that

$$v\hat{G} = [v\,G_1 \mid v\,G_2 \mid \ \ldots \ \mid v\,G_r].$$

Hence $p = \mathrm{wt}(v\,\hat{G}) = \mathrm{wt}(v\,G_1) + \mathrm{wt}(v\,G_2) + \ldots + \mathrm{wt}(v\,G_r)$.

Now we may also consider $v \in R^k$. Then since v and G_i have all entries in $\mathbb{F}$, we have

$$\begin{aligned} \mathrm{wt}(v\,G) &= \mathrm{wt}(v\,G_1\,\mathbf{z}^{\alpha_1} + \ldots + v\,G_r\,\mathbf{z}^{\alpha_r}) \\ &= \mathrm{wt}(v\,G_1\,\mathbf{z}^{\alpha_1}) + \ldots + \mathrm{wt}(v\,G_r\,\mathbf{z}^{\alpha_r}) \\ &= \mathrm{wt}(v\,G_1) + \ldots + \mathrm{wt}(v\,G_r) \\ &= p. \end{aligned}$$

Now C has a nonzero codeword of weight $\mathrm{dist}(C)$, and so $\mathcal{C}$ also has a codeword of weight $\mathrm{dist}(C)$. The lemma follows immediately from this. □

The following corollary gives what we will call *the block code bound.*

Corollary 2.5.2 (The block code bound.) *Suppose G is as in the proposition above. Suppose every block code of rate $\frac{k}{rn}$ has distance less than or equal to d_0. Then $\mathcal{C} = \mathcal{C}(G)$ has distance less than or equal to d_0.*

We now state a simple lemma which will lead to an example of the above ideas.

Lemma 2.5.3 *Any binary linear block code, C, of rate $\frac{2}{3r}$ (r a positive integer) has distance at most $2r$.*

Proof: We fix one of the $3r$ coordinates of C and look at the entries of the four codewords of C in this coordinate. By linearity either all codewords have a 0 in this coordinate, or two codewords have a 0 and two have a 1 in this coordinate. Hence the total number of 1's in all four codewords together is at most $2 \cdot 3r = 6r$. Of course the zero codeword has no 1's in it. So by a variant of the pigeonhole principle, not all of the three nonzero codewords can have weight more than $2r$. Hence $\mathrm{dist}(C) \leq 2r$. □

Example 2.5.4 Suppose $G \in R^{2\times 3}$ is a polynomial matrix that can be written using just r different monomials. Then $\mathrm{dist}(\mathcal{C}(G)) \leq 2r$.

We will come back to this example later (see Example 4.2.6), showing that the distance upper bound of $2r$ can be reached by a code with a generator matrix of polynomials of degree at most 1, provided $m \geq r - 1$.

2.6 Multidimensional Systems

In this section we look at multidimensional behavioral systems and consider their relationship to multidimensional convolutional codes.

The duality between modules and systems we describe below was developed in a very general context by Oberst [33]. In that paper, Oberst specifically considered the situation we look at here as a special case of his general theory. Our development here concentrates only on the situation where the ground ring is the polynomial ring $R = \mathbb{F}[z_1, ..., z_m]$, thus simplifying some of the arguments given by Oberst.

A nice development of this material may also be found in the preprint [53] of Wood, Rogers, and Owens. Here we take a somewhat different, perhaps simpler, approach.

This author along with Peterson and Rosenthal also considered some aspects of the module-system duality in [34].

In this section, we use the notation $\hat{R}$ for the power series ring $\mathbb{F}[[z_1, ..., z_m]]$. Also if v is a polynomial or a power series in the indeterminates $z_1, ..., z_m$, then for $\alpha \in \mathbb{N}^m$ we denote the $\mathbf{z}^\alpha$-coefficient of v by v_α. So $v_\alpha \in \mathbb{F}$, and

$$v = \sum_{\alpha \in \mathbb{N}^m} v_\alpha \mathbf{z}^\alpha.$$

Also if $v = [v_1 \ ... \ v_n]$ is an n-tuple of polynomials or power series, we will use the notation $v_{i,\alpha}$ to denote the $\mathbf{z}^\alpha$-coefficient of v_i.

Recalling the m-D finite sequence space $\mathcal{S}$ from Section 2.1, we now define the m-D sequence space, $\mathcal{S}_\infty$, as the space of all m-D sequences on $\mathbb{F}$. That is,

$$\mathcal{S}_\infty = \{w : \mathbb{N}^m \longrightarrow \mathbb{F}\}.$$

$\mathcal{S}_\infty$ differs from $\mathcal{S}$ in that the finite support requirement is dropped.

As we mentioned above, the systems we will be considering are behavioral systems. We give a general definition here, although later we will restrict our attention to a subclass of all behavioral systems.

Definition 2.6.1 A *general behavioral system of length* n (or *general behavior of length* n) is a subset $B \subseteq \mathcal{S}_\infty^n$. Elements of B are called *trajectories* of the system.

The behavioral approach to systems theory was pioneered by J. C. Willems beginning in the mid 1980's (see [50, 51, 52]). With this point of view, the emphasis shifts from the laws governing a system to the trajectories of the system itself. From the trajectories, information about the laws may be obtained (for instance see [49]).

Just as we identify $\mathcal{S}$ with the polynomial ring $R = \mathbb{F}[z_1, ..., z_m]$, so also we may identify $\mathcal{S}_\infty$ with the power series ring $\hat{R} = \mathbb{F}[[z_1, ..., z_m]]$ by means of the $\mathbb{F}$-isomorphism

$$\begin{array}{ccc} \mathcal{S}_\infty & \longrightarrow & \hat{R} \\ w & \longmapsto & \displaystyle\sum_{\alpha \in \mathbb{N}^m} w(\alpha)\mathbf{z}^\alpha. \end{array}$$

This isomorphism may be extended to an $\mathbb{F}$-isomorphism $\mathcal{S}_\infty^n \longrightarrow \hat{R}^n$ componentwise. With this isomorphism we may consider general behaviors of length n as subsets of $\hat{R}^n$.

It is well known that $\hat{R}^n \cong \hom_{\mathbb{F}}(R^n, \mathbb{F})$. We give an explicit $\mathbb{F}$-isomorphism as we will need it later. Let $\varphi \in \hom_{\mathbb{F}}(R^n, \mathbb{F})$. For $1 \leq i \leq n$ and for $\alpha \in \mathbb{N}^m$ define

$$\begin{array}{rcl} w_{\varphi,i,\alpha} & = & \varphi([0 \; ... \; 0 \;\; \mathbf{z}^\alpha \;\; 0 \; ... \; 0]) \in \mathbb{F} \qquad (\mathbf{z}^\alpha \text{ in the } i^{th} \text{ position}); \\ w_{\varphi,i} & = & \displaystyle\sum_{\alpha \in \mathbb{N}^m} w_{\varphi,i,\alpha}\mathbf{z}^\alpha \in \hat{R}; \\ w_\varphi & = & [w_{\varphi,1} \;\; w_{\varphi,2} \; ... \; w_{\varphi,n}] \in \hat{R}^n. \end{array}$$

The correspondence

$$\begin{array}{ccc} \hom_{\mathbb{F}}(R^n, \mathbb{F}) & \longrightarrow & \hat{R}^n \\ \varphi & \longmapsto & w_\varphi \end{array}$$ is the desired $\mathbb{F}$-isomorphism.

In order to clearly see the connection between $\varphi \in \hom_{\mathbb{F}}(R^n, \mathbb{F})$ and $w_\varphi \in \hat{R}^n$ we introduce an important nondegenerate bilinear form

$$\langle \;\, , \;\, \rangle : \hat{R}^n \times R^n \longrightarrow \mathbb{F}.$$

For $w = [w_1 \; ... \; w_n] \in \hat{R}^n$ and $v = [v_1 \; ... \; v_n] \in R^n$, we define

$$\langle w, v \rangle := \sum_{i=1}^{n} \sum_{\alpha \in \mathbb{N}^m} w_{i,\alpha} v_{i,\alpha} \in \mathbb{F}.$$

Observe that the sum used in defining $\langle w, v \rangle$ is well defined, for all but finitely many terms are 0 since v is supported finitely on $\mathbb{N}^m$. This bilinear form may be thought of as a dot product of an element of $\mathcal{S}_\infty^n$ with an element of $\mathcal{S}^n$.

We note that this bilinear form was first introduced in a different context by Macaulay [27] in 1916. It is also used by Oberst in the paper [33, pp. 51, 52] cited above. In the 1-D case it is used by Rosenthal, Schumacher, and York [43] and by York [55].

The relationship between $\varphi \in \hom_{\mathbb{F}}(R^n, \mathbb{F})$ and $w_\varphi \in \hat{R}^n$ may be expressed using this bilinear form. In fact for any $v \in R^n$,

$$\varphi(v) = \langle w_\varphi, v \rangle.$$

Remark 2.6.2 We may consider the bilinear form $\langle\ ,\ \rangle$ as being defined for any value of n, and in particular for $n=1$. Then for $w=[w_1\ ...\ w_n]\in\hat{R}^n$ and $v=[v_1\ ...\ v_n]\in R^n$ we have $\langle w,v\rangle=\sum_{i=1}^{n}\langle w_i,v_i\rangle$

Next we introduce the left-shift operators on $\mathcal{S}_\infty$. For $1\le j\le m$ we define the shift operator σ_j as a map

$$\begin{array}{rcl}\sigma_j:\mathcal{S}_\infty & \longrightarrow & \mathcal{S}_\infty\\ w & \longmapsto & w\,\sigma_j\\ & & (w\,\sigma_j)(i_1,...,i_m)=w(i_1,...,i_{j-1},i_j+1,i_{j+1},...,i_m)\end{array}$$

Remark 2.6.3

i) If we identify $\mathcal{S}_\infty$ with $\hat{R}$ as above, then the effect of applying σ_j to $w\in\hat{R}$ is to calculate $z_j^{-1}w$, discarding any terms in which the exponent on z_j is negative. We refer to this operation on $\hat{R}$ as left shift with truncation. (This may be made mathematically precise by considering working in the quotient ring $\mathbb{F}((z_1,z_1^{-1},\ldots,z_m,z_m^{-1}))/(z_1^{-1},\ldots,z_m^{-1})$ where $\mathbb{F}((z_1,z_1^{-1},\ldots,z_m,z_m^{-1}))$ is the field of Laurent series in m indeterminates; i.e., these series have left-bounded support. We will not pursue this point of view here. In the 1-D case this situation is considered by Piret [35].)

ii) The maps σ_j , $1\le j\le m$ allow us to let any monomial in σ_1 , ..., σ_m act on $\mathcal{S}_\infty$ by composition (the order of factors is immaterial–the σ_j's commute). The corresponding action of a monomial $\sigma_1^{i_1}\cdots\sigma_m^{i_m}$ on an element $w\in\hat{R}^n$ is multiplication $w\cdot(z_1^{-i_1}\cdots z_m^{-i_m})$ with truncation (discarding any term in which any negative exponent appears in the monomial part of the term). We may shorten all this using multi-index notation: $w\,\underline{\sigma}^\alpha=w\,\mathbf{z}^{-\alpha}$ with truncation. where $\alpha\in\mathbb{N}^m$, and $\underline{\sigma}^\alpha=\sigma_1^{\alpha_1}\ \cdots\ \sigma_m^{\alpha_m}$

iii) The action in ii) may be extended $\mathbb{F}$-linearly so that every polynomial in $\mathbb{F}[\sigma_1,...,\sigma_m]$ acts on $\hat{R}$.

For example, using $\mathbb{F}_2$ as the ground field,

$$\begin{aligned}&(1+z_1+z_1z_2+z_1z_2^3+\cdots)(1+\sigma_1+\sigma_1\sigma_2)\\ &\qquad=1+(z_1+1)+(z_1z_2+z_2+1)+(z_1z_2^3+z_2^3+z_2^2)+\cdots.\end{aligned}$$

iv) σ_j and, by the extensions in ii) and iii), any polynomial in $\mathbb{F}[\sigma_1,...,\sigma_m]$ acts on $\hat{R}^n$ by acting on each component individually. That is,

$$[w_1\ \ w_2\ ...\ w_n]r(\sigma)=[w_1r(\sigma)\ \ w_2r(\sigma)\ ...\ w_nr(\sigma)].$$

Here is a lemma giving some of the properties of the left-shift operators and the bilinear form $\langle\ ,\ \rangle$.

Lemma 2.6.4 *Let $p\in\hat{R}$, $r\in R$, $w=[w_1\ ...\ w_n]\in\hat{R}^n$, and $v=[v_1\ ...\ v_n]\in R^n$.*
(i) For any $\alpha,\ \beta\in\mathbb{N}^m$, $(p\,\underline{\sigma}^\beta)_\alpha=p_{\alpha+\beta}\in\mathbb{F}$.

(ii) For any $\beta\in\mathbb{N}^m$, the map $\underline{\sigma}^\beta:\hat{R}^n\longrightarrow\hat{R}^n$ is dual to the multiplication map $\mathbf{z}^\beta:R^n\longrightarrow R^n$ with respect to the bilinear form $\langle\ ,\ \rangle$. (Duality of maps with respect to a bilinear form is discussed in Greub [16, pp. 67-73].) That is,

$$\langle w,\,\mathbf{z}^\beta\,v\rangle=\langle w\,\underline{\sigma}^\beta,v\rangle.$$

(iii) For any $\beta\in\mathbb{N}^m$, $(p\,r(\underline{\sigma}))_\beta=\langle p\,\underline{\sigma}^\beta,r\rangle\in\mathbb{F}$.

Proof:
(i) $p\,\underline{\sigma}^\beta=p\,\mathbf{z}^{-\beta}$ with truncation. The z^α-term of this product is $p_{\alpha+\beta}\,\mathbf{z}^{\alpha+\beta}\,\mathbf{z}^{-\beta}=p_{\alpha+\beta}\,\mathbf{z}^\alpha$.

(ii) We have

$$\begin{aligned}\langle w,\,\mathbf{z}^\beta\,v\rangle &= \sum_{i=1}^{n}\sum_{\alpha\in\mathbb{N}^m}w_{i,\alpha}\,(\mathbf{z}^\beta\,v_i)_\alpha = \sum_{i=1}^{n}\sum_{\substack{\alpha\in\mathbb{N}^m:\\ \alpha-\beta\in\mathbb{N}^m}}w_{i,\alpha}v_{i,\alpha-\beta}\\ &= \sum_{i=1}^{n}\sum_{\alpha\in\mathbb{N}^m}w_{i,\alpha+\beta}\,v_{i,\alpha} = \sum_{i=1}^{n}\sum_{\alpha\in\mathbb{N}^m}(w_i\,\underline{\sigma}^\beta)_\alpha\,v_{i,\alpha}\\ &= \langle w\,\underline{\sigma}^\beta,v\rangle.\end{aligned}$$

(iii) We have

$$(p\,r(\underline{\sigma}))_\beta = \sum_{\alpha\in\mathbb{N}^m} p_{\alpha+\beta}\, r_\alpha = \sum_{\alpha\in\mathbb{N}^m} (p\,\underline{\sigma}^\beta)_\alpha r_\alpha = \langle p\,\underline{\sigma}^\beta, r\rangle.$$

□

Definition 2.6.5 Let $G \in R^{l\times n}$ be any polynomial matrix. Let $G(\underline{\sigma})$ be the matrix obtained from G by replacing z_j by σ_j for $1 \le j \le m$. (Sometimes we may write $G(\mathbf{z})$ to emphasize the distinction). We define a general behavior $\mathcal{B}(G) \subseteq \hat{R}^n$ by

$$\mathcal{B}(G) = \ker G^t(\underline{\sigma}) = \{w \in \hat{R}^n \ :\ w\, G^t(\underline{\sigma}) = 0 \in \hat{R}^l\}.$$

A general behavior defined in this manner (as the kernel in $\hat{R}^n$ of the transpose of a polynomial matrix with n columns) is an *autoregressive behavioral system of length* n (or an *autoregressive behavior of length* n).

From now on all behaviors considered will be autoregressive behaviors. Hence we shall frequently drop the qualifier 'autoregressive' and simply refer to a behavior. We may also refer to a behavior as a *system.*

We are now ready to describe the duality between modules and behaviors.

Let $M \subseteq R^n$ be an R-submodule of the free module R^n. There is an exact sequence of R-modules:

$$0 \longrightarrow M \longrightarrow R^n \longrightarrow R^n/M \longrightarrow 0.$$

We apply the left-exact contravariant functor $\hom_{\mathbb{F}}(\underline{\quad}, \mathbb{F})$ (see for instance [18, p.200]) to this exact sequence. Remember that $\hom_{\mathbb{F}}(R^n, \mathbb{F}) \cong \hat{R}^n$. We obtain

$$0 \longrightarrow \hom_{\mathbb{F}}(R^n/M, \mathbb{F}) \longrightarrow \hat{R}^n \longrightarrow \hom_{\mathbb{F}}(M, \mathbb{F}).$$

So we may think of $\hom_{\mathbb{F}}(R^n/M, \mathbb{F}) \subseteq \hat{R}^n \cong \hom_{\mathbb{F}}(R^n, \mathbb{F})$. Also $\hom_{\mathbb{F}}(R^n/M, \mathbb{F})$ is the linear algebra dual of R^n/M. Define

$$B(M) = \{\varphi \in \hom_{\mathbb{F}}(R^n, \mathbb{F}) \ :\ M \subseteq \ker\varphi\}.$$

Using standard arguments, it follows at once that $B(M) \cong \hom_{\mathbb{F}}(R^n/M, \mathbb{F})$. Hence we may also regard $B(M)$ as the linear algebra dual of R^n/M.

For the next proposition recall the isomorphism $\varphi \longmapsto w_\varphi$ between $\hom_{\mathbb{F}}(R^n, \mathbb{F})$ and $\hat{R}^n$.

Proposition 2.6.6 *Let $G \in R^{l\times n}$ be a polynomial matrix. Let $M = \operatorname{rowspace} G \subseteq R^n$. As above, let $B(M) = \{\varphi \in \hom_F(R^n, \mathbb{F}) :\ M \subseteq \ker\varphi\}$. Then the behavior $\mathcal{B}(G) = \{w_\varphi \in \hat{R}^n \ :\ \varphi \in B(M)\}$.*

Proof: In the chain of equivalences that follows, the fourth and sixth equivalences follow from Lemma 2.6.5 parts (ii) and (iii) respectively. We have

$$\begin{array}{rcll}
\varphi \in B(M) & \Leftrightarrow & \varphi(vG) = 0 & \text{for all } v \in R^l \\
& \Leftrightarrow & \langle w_\varphi, vG\rangle = 0 & \text{for all } v \\
& \Leftrightarrow & \langle w_\varphi, \mathbf{z}^\beta g\rangle = 0 & \text{for all } \beta \in \mathbb{N}^m \text{ and for all rows} \\
& & & g = [g_1 \ldots g_n] \text{ of } G \text{ (by} \\
& & & \mathbb{F}\text{-bilinearity of } \langle\ ,\ \rangle) \\
& \Leftrightarrow & \langle w_\varphi\, \underline{\sigma}^\beta, g\rangle = 0 & \text{for all } \beta,\ g \\
& \Leftrightarrow & \displaystyle\sum_{i=1}^n \langle w_{\varphi,i}\, \underline{\sigma}^\beta, g_i\rangle = 0 & \text{for all } \beta,\ g \\
& \Leftrightarrow & \left(\displaystyle\sum_{i=1}^n w_{\varphi,i}\ g_i(\underline{\sigma})\right)_\beta = 0 & \text{for all } \beta,\ g \\
& \Leftrightarrow & \displaystyle\sum_{i=1}^n w_{\varphi,i}\ g_i(\underline{\sigma}) = 0 & \text{for all } g \\
& \Leftrightarrow & w_\varphi \in \mathcal{B}(G). &
\end{array}$$

The desired result follows immediately from this. $\square$

Remark 2.6.7 The above proposition may be interpreted as saying that $B(M) \cong \mathcal{B}(G)$ because of the isomorphism $\varphi \longmapsto w_\varphi$. Hence $\mathcal{B}(G)$ may be regarded as the linear algebra dual of R^n/M.

This establishes a linear algebra duality between finitely generated R-modules (any of which may be expressed as $R^n/\text{rowspace}(G)$ for an appropriate polynomial matrix G) and behaviors of the form $\mathcal{B}(G)$.

Oberst [33, p.3, p.34] shows considerably more, namely that this duality between modules and systems is a categorical duality. This means that any statement about modules has a corresponding systems theoretic statement and vice versa.

Another important result from Oberst [33, pp. 61-62] is a characterization of those general behaviors that are autoregressive behaviors. We state this result of Oberst here. A discussion of the linearly compact topology appearing in the proposition may be found in Köthe [22, p. 95]. For our discussion, one may think of a subset of $\mathcal{S}_\infty^n$ as being closed if it is closed under the operation of taking infinite sums for which given a bounded subset of $\mathbb{N}^m$, only finitely many of the terms of the sum have support intersecting the given bounded subset.

Proposition 2.6.8 *A general behavior $B \subseteq \hat{R}^n$ is an autoregressive behavior if and only if B is $\mathbb{F}$-linear, left-shift invariant, and closed in the linearly compact topology of $\hat{R}^n$.*

Proofs of some special cases of the previous proposition may be found separately in the literature. The 1-D case may be found in Willems [50]. The 2-D case may be found in Rocha [38, p.15, p.150] and in Rocha and Willems [39].

The following is an example of a behavior that is $\mathbb{F}$-linear, left-shift invariant, but not closed in the linearly compact topology of $\hat{R}^n$.

Example 2.6.9 The set of all finite support sequences in $\mathcal{S}_\infty^n$ is linear and left-shift invariant, but it is not closed (in fact this set is dense in $\mathcal{S}_\infty^n$). This set corresponds to the set of polynomials in $\hat{R}^n$.

So by Proposition 2.6.8, this set is not an autoregressive behavior. That is, it cannot be represented as a kernel of a polynomial matrix with respect to the left-shift action of R on $\hat{R}$.

Remark 2.6.10 As mentioned at the beginning of this section, the module-system duality has been shown by Oberst in [33] to apply to many rings. In particular the duality applies to the Laurent polynomial ring $\mathbb{F}[z_1, z_1^{-1}, \ldots, z_m, z_m^{-1}]$. In this case behaviors are supported on all of $\mathbb{Z}^m$ (in contrast to the situation studied here where behaviors are supported on $\mathbb{N}^m$). This is the approach taken by Fornasini and Valcher in their work [9, 10, 11, 8, 47]. One advantage to the Laurent polynomial approach is that then both on the module side and on the behavior side, instead of one-sided shift invariance (which occurs in the polynomial case studied in this dissertation), we have two-sided shift invariance.

CHAPTER 3

MATRICES AND MODULES

Although we are mainly interested in matrices and modules over the polynomial ring $R = \mathbb{F}[z_1, ..., z_m]$, the results are of general interest, and so we have put some of them in a broader setting. We do assume that all rings are rings with identity.

3.1 Matrix Primeness

Let D be a noetherian unique factorization domain (UFD). Let $G \in D^{k \times n}$, $k \leq n$, be a matrix with entries in D. Suppose G has a nonzero $k \times k$ minor. Let $S \subseteq D$ be the set of full-size minors of G. Let $J = (S)$ be the ideal in D generated by S.

Definition 3.1.1 In the situation above, the matrix G is

i) *zero prime (ZP)* if $J = D$;

ii) *minor prime (MP)* if the elements of S have no common divisors in D except for units;

iii) *left-factor prime (LFP)* if whenever G is factored as a product $G = T \cdot \bar{G}$ with $T \in D^{k \times k}$ and $\bar{G} \in D^{k \times n}$, we have that $\det(T)$ is a unit of D (i.e., T is unimodular).

The following two propositions are well known. We include the proofs for completeness.

Proposition 3.1.2 *ZP $\Rightarrow$ MP $\Rightarrow$ LFP.*

Proof: ZP $\Rightarrow$ MP: Suppose G is ZP. Let f be a common factor of the full-size minors of G. We have $J \subseteq (f)$, but also $J = D$ since G is ZP. So $D = (f)$. So f is a unit. Hence G is MP.

MP $\Rightarrow$ LFP: Suppose G is MP. Write $G = T \cdot \bar{G}$ where $T \in D^{k \times k}$ with nonzero determinant. Now the determinant of T divides every full size minor of G (for any ($k \times k$ submatrix of G)= $T\cdot$ (corresponding $k \times k$ submatrix of $\bar{G}$)). Hence $\det(T)$ is a unit since the full-size minors of G have no nonunit common factors. Conclude that G is LFP. $\square$

Proposition 3.1.3 *If D is a principal ideal domain (PID), then LFP $\Rightarrow$ ZP.*

Proof: Suppose G is not ZP. So $J \neq D$. Since D is a PID, $J = (f)$ for some $f \in D$ where f is nonzero and a nonunit (for if f is a unit, then $J = D$, contradicting that G is not ZP). By the Smith normal form [20, p.176] for G, there exist matrices $U \in D^{k \times k}$ and $V \in D^{n \times n}$ such that $\det(U)$ and $\det(V)$ are units in D and such that

$$U \cdot G \cdot V = \left[\begin{array}{ccc|c} d_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & 0 \\ 0 & \cdots & d_k & \end{array}\right] \text{ where } d_1 d_2 ... d_k = f.$$

Then

$$G = \left(U^{-1} \cdot \begin{bmatrix} d_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_k \end{bmatrix}\right) \cdot \left(\left[\begin{array}{ccc|c} 1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & 0 \\ 0 & \cdots & 1 & \end{array}\right] \cdot V^{-1}\right).$$

This factorization of G shows that G is not LFP, for

$$\det\left(U^{-1}\cdot\begin{bmatrix} d_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_k \end{bmatrix}\right) = \det(U^{-1})\cdot f,$$

which is not a unit in D. □

The results until the end of this section are given in the context of the polynomial ring $\mathbb{F}[z_1, ..., z_m]$.

Corollary 3.1.4 *If $m = 1$, then ZP, MP, and LFP are all equivalent.*

Proof: A polynomial ring in 1 indeterminate over a field is a PID. □

Morf, Levy, and Kung [31] prove the following:

Proposition 3.1.5 *If $m = 2$ then MP $\Leftrightarrow$ LFP.*

Remark 3.1.6 Observe that $G = \begin{bmatrix} z_1 & z_2 \end{bmatrix}$ shows that for $m = 2$, ZP differs from MP and LFP.

The inequivalence of MP and LFP for $m \geq 3$ was first observed by Youla and Gnavi [56]. The following proposition is the same as an example given by Zerz [59] to demonstrate this inequivalence.

Proposition 3.1.7 *If $m \geq 3$, then LFP does not imply MP.*

Proof: Let $G = \begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{bmatrix}$ over the ring $R = \mathbb{F}[z_1, z_2, z_3]$ ($\mathbb{F}$ any field). The full-size minors of G have a common factor of z_1, so G is not MP.

We show that G is LFP. Suppose to the contrary that G is not LFP. Then we can factor $G = T \cdot \bar{G}$ where $T \in R^{2\times 2}$, and $\bar{G} \in R^{2\times 3}$ where $\det(T)$ is a nonzero nonunit of R. However $\det(T)$ must divide any common factor of the full-size minors of G. Hence we must have $\det(T) = z_1$. Then also $\bar{G}$ is LFP.

Now let $H = \begin{bmatrix} z_2 & z_3 & -z_1 \end{bmatrix}$. Observe that $G \cdot H^t = \begin{bmatrix} 0 & 0 \end{bmatrix}^t$. But then we must also have $\bar{G} \cdot H^t = \begin{bmatrix} 0 & 0 \end{bmatrix}^t$. So all the entries of $\bar{G}$ must have zero constant term. From this it follows that every full-size minor of $\bar{G}$ is of degree at least 2. Then the full-size minors of $T \cdot \bar{G}$ are of degree of at least 3. But $T \cdot \bar{G} = G$ which has all full-size minors of degree 2. This is a contradiction, and so we conclude that G is LFP. □

We summarize these results as follows:

Proposition 3.1.8

i) If $m = 1$, then ZP, MP, and LFP are all equivalent.

ii) If $m = 2$, then MP and LFP are equivalent; ZP is different.

iii) If $m \geq 3$, then all three are different.

3.2 Determinantal Factorization

The preceding section raises some questions that, while somewhat tangential to the main flow of this paper, are quite interesting. In order to state the questions succinctly, we introduce some terminology.

Definition 3.2.1 Let D be a noetherian UFD. Let $G \in D^{k\times n}$ ($k \leq n$). Let $d \in D$ be a common factor of all the $k \times k$ minors of G. A *determinantal factorization of G with respect to d* is a factorization $G = T \cdot \bar{G}$ where $T \in D^{k\times k}$, $\bar{G} \in D^{k\times n}$, and $\det T = d$.

We will call a matrix *wide* if it has at least as many columns as rows. Suppose D has the property that for any wide matrix G and for any common factor, d, of the full-size minors of G, there is a determinantal factorization of G with respect to d. Then we say D is a *determinantal factorization domain* (DFD).

In other words, a DFD is a noetherian UFD in which left factor primeness of a matrix implies minor primeness of that matrix.

If D has the property that it admits determinantal factorizations for square matrices (but not necessarily for general wide matrices), then we will call D a DFD for square matrices.

Certain results in the literature (to be cited below) are stated and proved just for square matrices. So we begin by establishing that a DFD for square matrices is a DFD. We do this in several steps. First a lemma.

Lemma 3.2.2 *Let D be a UFD. Let $G \in D^{k\times k}$ and $v \in D^k$. Let G_j be the $k \times k$ matrix obtained from G by replacing the j^{th} column of G by v^t. Then*

$$(\text{adj } G) \cdot v^t = \begin{bmatrix} \det G_1 & \det G_2 & \dots & \det G_k \end{bmatrix}^t .$$

Proof: Let a_{ij} be the ij^{th} entry of adj(G). Let b_{ij} be the determinant of the $(k-1) \times (k-1)$ submatrix of G obtained by deleting the i^{th} row and j^{th} column of G. So $a_{ij} = (-1)^{i+j} b_{ji}$.

For $1 \le j \le k$ we expand $\det(G_j)$ along the j^{th} column. We obtain

$$\det\,(G_j) = \sum_{i=1}^{k} v_i \cdot (-1)^{i+j} b_{ij} = \sum_{i=1}^{k} v_i \cdot a_{ji} = ((\text{adj } G) \cdot v^t)_j \,.$$

□

At this point we cannot resist giving the following simple proof of Cramer's rule.

Corollary 3.2.3 *Cramer's rule.*

Proof: Consider the equation $G \cdot x^t = v^t$ ($\det(G) \neq 0$). Then $(\det G) \cdot x^t = (\text{adj } G) \cdot v^t$. Hence

$$x^t = \frac{1}{\det G} \cdot \begin{bmatrix} \det G_1 & \det G_2 & \dots & \det G_k \end{bmatrix}^t .$$

□

Lemma 3.2.4 *Let D be a DFD for square matrices. Let $G \in D^{k\times n}$ ($k \le n$), and suppose G has a nonzero full-size minor. Suppose d is an irreducible element of D such that d divides every full-size minor of G. Then G can be factored as $G = T_d G_d$ where $T_d \in D^{k\times k}$ with $\det(T_d) = d$, and $G_d \in D^{k\times n}$.*

Proof: Since G has a nonzero full-size minor, there exists some power d^q of d that does not divide every full-size minor of G. Let d^p be the highest power of d that divides every full-size minor of G (by hypothesis, $p \ge 1$). So there is a full-size minor of G that is not divisible by d^{p+1}. Without loss of generality we may assume that the minor corresponding to the first k columns of G is not divisible by d^{p+1}.

That is $G = [A \mid B]$ where $A \in D^{k\times k}$ with $\det(A)$ not divisible by d^{p+1}.

Now since D is a DFD for square matrices, there exists $\tilde{T_d} \in D^{k\times k}$ and $A_d \in D^{k\times k}$ such that $\det(\tilde{T_d}) = d^p$ and $A = \tilde{T_d} A_d$. Let $f = \det(A_d)$. Since d^p is the highest power of d dividing $\det(A)$ we have that d does not divide f.

Now by Lemma 3.2.2, every entry of $(\text{adj } A)B$ is, up to sign, a full-size minor of G. Therefore d^p divides every entry of $(\text{adj } A)B = (\text{adj } A_d)(\text{adj } \tilde{T_d})B$. So d^p divides every entry of $A_d(\text{adj } A_d)(\text{adj } \tilde{T_d})B = f(\text{adj } \tilde{T_d})B$.

Now d and f have no nonunit common factors since d is irreducible and d does not divide f. Therefore d^p divides every entry of $(\text{adj } \tilde{T_d})B$. But then in the field of quotients of D, we have

$$\tilde{T_d}^{-1} B = \frac{\text{adj } \tilde{T_d}}{\det \tilde{T_d}} B = \frac{1}{d^p}(\text{adj } \tilde{T_d}\ B)$$

has all its entries in D.

Put $B_d = \tilde{T_d}^{-1} B$. We then have that

$$G = [A \mid B] = [\tilde{T_d} A_d \mid \tilde{T_d} B_d] = \tilde{T_d}[A_d \mid B_d].$$

Now $\tilde{T_d}$ has determinant d^p and is square. Since D is a DFD for square matrices, we may factor $\tilde{T_d} = T_d\, T'$ where $T_d, T' \in D^{k\times k}$ with $\det(T_d) = d$. We then put $G_d = T'\ [A_d \mid B_d]$. This gives $G = T_d\ G_d$ where $T_d \in D^{k\times k}, G_d \in D^{k\times n}$, and $\det(T_d) = d$. □

Corollary 3.2.5 *Let D be a DFD for square matrices. Suppose $G \in D^{k\times n}$ and all full-size minors of G are zero. Let d be an irreducible element of D. Then we may factor G as $G = T_d\, G_d$ where $T_d \in D^{k\times k}, G_d \in D^{k\times n}$, and $\det(T_d) = d$.*

Proof: Let $G_1 = [d\ I_k \mid G] \in D^{k\times(k+n)}$. Now the full-size minors of G_1 are all divisible by d. Apply Lemma 3.2.4 to G_1 obtaining a factorization $G_1 = T_d\, G_{1,d}$ where $T_d \in D^{k\times k}, G_{1,d} \in D^{k\times(k+n)}$, and $\det(T_d) = d$. Partition $G_{1,d} = [C \mid G_d]$ where $C \in D^{k\times k}$, and $G_d \in D^{k\times n}$. Then we have

$$[d\ I_k \mid G] = G_1 = T_d\, [C \mid G_d],$$

from which we obtain $G = T_d\, G_d$, the desired factorization. □

Proposition 3.2.6 (DFD for square matrices ⇒ DFD) *If D is a DFD for square matrices, then D is a DFD.*

Proof: Let $G \in D^{k\times n}$ have all its full-size minors divisible by an element $g \in D$. We must show that G may be factored as $G = T\,\bar{G}$ where $T \in D^{k\times k}, \bar{G} \in D^{k\times n}$, and $\det(T) = g$. Write $g = d_1 \cdot d_2 \cdot \ldots \cdot d_r$ where each d_i is an irreducible element of D (the d_i's need not be distinct).

We proceed by induction on r. The case of $r = 0$, corresponding to g being a unit in D, is trivial. We assume a determinantal factorization is possible with respect to any common factor of the full-size minors of G having at most $r-1$ irreducible factors ($r \geq 1$).

With this induction hypothesis we write $G = T_1\, G_1$ where $T_1 \in D^{k\times k}$, and $\det(T_1) = d_1 \cdot d_2 \cdot \ldots \cdot d_{r-1}$. But then any full-size minor of G_1 is divisible by d_r (since any full-size minor of G_1 is equal to the corresponding full-size minor of G divided by $d_1 \cdot d_2 \cdot \ldots \cdot d_{r-1}$). We then apply Lemma 3.2.4 or Corollary 3.2.5 (according to whether the full-size minors of G_1 are not or are all zero) to write $G_1 = T_2\,\bar{G}$ where $T_2 \in D^{k\times k}$ with $\det(T_2) = d_r$. Then put $T = T_1\, T_2$, which gives

$$G = T_1\, G_1 = T_1\, T_2\, \bar{G} = T\,\bar{G}.$$

And $\det(T) = (\det T_1)(\det T_2) = (d_1 \cdot d_2 \cdot \ldots \cdot d_{r-1})d_r = g$.

We conclude that D is a DFD, as desired. □

We now ask what noetherian UFDs are DFDs, and we also consider some methods for obtaining determinantal factorizations.

Estes and Matijevic [6] prove that a ring is a DFD for square matrices if and only if it is a Towber ring. In an earlier paper [5], the same authors prove that if D is a Dedekind domain, then the polynomial ring $D[z]$ in one indeterminate is a DFD for square matrices. It is well known (see for instance [19]) that any PID is a Dedekind domain. Hence if D is a PID, then $D[z]$ is a DFD for square matrices. What is especially interesting about this weaker result is that the converse is also true. The key example for proving the converse is also given in [5], but for a different purpose. We give the details in the following proposition. This proposition is appealing for two reasons. First, it gives a reason why for $m \leq 2$ we have minor primeness and left factor primeness equivalent in $\mathbb{F}[z_1, ..., z_m]$, and why this equivalence fails for larger m. Second, it shows that a polynomial ring in one indeterminate over a PID is better than a polynomial ring in one indeterminate over a non-principal UFD, even though the standard theorem [19, Theorem 16.12] states that each is simply a UFD.

Proposition 3.2.7 *Let D be a noetherian UFD. $D[z]$ is a DFD if and only if D is a PID.*

Proof: From the above discussion we only need prove that if $D[z]$ is a DFD, then D is a PID. We proceed by contrapositive.

Suppose D is not a PID. Choose $f, g \in D$ such that the ideal (f, g) is not principal (such f, g exist because D is noetherian). We may assume that f and g have no nonunit common factors (we can cancel such if necessary). Observe that $1 \notin (f, g)$ for otherwise we would have $(f, g) = (1) = D$ and then (f, g) would be principal.

Now put $G = \begin{bmatrix} f^2 & fg+z \\ fg-z & g^2 \end{bmatrix} \in D[z]^{2\times 2}$. Note that $\det(G) = z^2$.

We now show that G has no factorization $G = T \cdot \bar{G}$ where $T, \bar{G} \in D[z]^{2\times 2}$ with $\det(T) = z$. This will complete the proof, for G will not have a determinantal factorization with respect to z.

Suppose to the contrary that there is such a factorization $G = T \cdot \bar{G}$. We may write

$$T = \begin{bmatrix} a_0 z^2 + a_1 z + a_2 & b_0 z^2 + b_1 z + b_2 \\ c_0 z^2 + c_1 z + c_2 & d_0 z^2 + d_1 z + d_2 \end{bmatrix}$$

where $a_0, b_0, c_0, d_0 \in D[z]$ and $a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2 \in D$.

Now $z = \det(T)$, but looking at the expression for T we see that the z term of $\det(T)$ is $(a_1 d_2 + a_2 d_1 - b_1 c_2 - b_2 c_1)z$. Therefore $a_1 d_2 + a_2 d_1 - b_1 c_2 - b_2 c_1 = 1$. Hence $1 \in (a_2, b_2, c_2, d_2)$ (considered as an ideal of D).

Now putting $a = a_0 z^2 + a_1 z + a_2, b = b_0 z^2 + b_1 z + b_2, c = c_0 z^2 + c_1 z + c_2,$ and $d = d_0 z^2 + d_1 z + d_2$ we have

$$\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \cdot T = \begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix}$$

since $\det(T) = z$.

Therefore

$$\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \cdot G = \begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} \cdot \bar{G}$$

since $G = T \cdot \bar{G}$.

In particular every entry of $\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \cdot G$ is divisible by z. So $df^2 - bfg + bz$ is divisible by z, and then so is $df - bg$. Hence $(d_0 z^2 + d_1 z + d_2)f - (b_0 z^2 + b_1 z + b_2)g$ is divisible by z. So $d_2 f - b_2 g$ is divisible by z. Since $f, g, b_2, d_2 \in D$, we must have $d_2 f = b_2 g$. Now since f, g have no nonunit common factors it follows that $f | b_2$ and $g | d_2$. Therefore $b_2, d_2 \in (f, g)$. A similar argument shows that $a_2, c_2 \in (f, g)$. Then $(a_2, b_2, c_2, d_2) \subseteq (f, g)$. But from above this gives $1 \in (f, g)$. This is a contradiction, and we conclude that G has no determinantal factorization with respect to z. $\square$

Example 3.2.8 Letting f and g be as in the example given in the above proposition, consider the matrix

$$G = \begin{bmatrix} z & f \\ 0 & g \end{bmatrix}.$$

Using arguments similar to those in the preceding proposition, one can show that G cannot be factored as $G = T \cdot \bar{G}$ where T is 2×2 and $\det(T) = z$. In this example note that even though such a factorization is impossible, the matrix is not left-factor prime, for the matrix $\begin{bmatrix} 1 & 0 \\ 0 & g \end{bmatrix}$ is a left factor of G with nonunit determinant.

Example 3.2.9 $\mathbb{Z}[x]$ the polynomial ring in one variable over the integers is a DFD.

3.2.1 Techniques for Determinantal Factorization

In this subsection we will consider methods for finding determinantal factorizations. Methods for obtaining determinantal factorizations in a polynomial ring in two variables over an algebraically closed field are also considered by Morf, Kung, and Levy in [31]. We begin with a simple but very useful lemma.

Lemma 3.2.10 *Let D be a commutative ring. Let $G \in D^{k\times n} (k \leq n)$. Suppose there is a unimodular matrix $U \in D^{k\times k}$ such that all the entries of the first row of $U G$ have a common factor $f \in D$. Then G can be factored as $G = T\,\bar{G}$ where $T \in D^{k\times k}$, $\bar{G} \in D^{k\times n}$, and $\det(T) = f$.*

Proof: We have

$$U\,G = \begin{bmatrix} f\,v \\ \hat{G} \end{bmatrix},$$

where $v \in D^n$ and $\hat{G} \in D^{(k-1)\times n}$. Hence

$$U\,G = \left[\begin{array}{c|ccc} f & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & I_{k-1} & \\ 0 & & & \end{array}\right] \left[\begin{array}{c} v \\ \hline \hat{G} \end{array}\right].$$

Then letting $u = \det(U)$, we have

$$G = U^{-1} \left[\begin{array}{c|ccc} u\,f & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & I_{k-1} & \\ 0 & & & \end{array}\right] \left[\begin{array}{c} u^{-1}\,v \\ \hline \hat{G} \end{array}\right].$$

Now let

$$T = U^{-1} \left[\begin{array}{c|ccc} u\,f & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & I_{k-1} & \\ 0 & & & \end{array}\right], \text{ and } \quad \bar{G} = \left[\begin{array}{c} u^{-1}\,v \\ \hline \hat{G} \end{array}\right].$$

Then $G = T\,\bar{G}$. Also $\det(T) = \det(U^{-1})\,u\,f = u^{-1}\,u\,f = f$. So we have the desired factorization. □

Now let D be a noetherian UFD. Suppose that $G \in D^{k\times n}$, and all the full-size minors of G have a common factor of f. We consider how we might find a unimodular matrix U such that $U\,G$ has its first row divisible by f. By Lemma 3.2.10 above, finding such a U will allow us to give a determinantal factorization of G with respect to f.

Consider performing elementary row operations on G (here elementary row operations are of three types: (i) multiply a row by a unit of D; (ii) exchange two rows; and (iii) replace a row by the sum of that row and a D-linear combination of the other rows). Of course any elementary row operation can be performed by left multiplication by a suitable unimodular matrix. In fact if we form the matrix $[I_k \mid G] \in D^{k\times(k+n)}$ and perform any sequence of elementary row operations on this augmented matrix to obtain $[U \mid \tilde{G}]$ where $U \in D^{k\times k}$ and $\tilde{G} \in D^{k\times n}$, then U is unimodular, and $U\,G = \tilde{G}$.

Combining this with Lemma 3.2.10 above, we see that if by applying elementary row operations to G we obtain a matrix where the first row is divisible by f, then G can be determinantally factored with respect to f.

Example 3.2.11 Let $\mathbb{F}$ be any field, and let $D = \mathbb{F}[x, y]$ be the polynomial ring in two indeterminates over $\mathbb{F}$. Let $G = \begin{bmatrix} x & y \\ y & x \end{bmatrix}$. So $\det(G) = x^2 - y^2 = (x + y)(x - y)$.

Suppose we want to factor $G = T\,\bar{G}$ where $T \in D^{2\times 2}$ and $\det(T) = x + y$. We observe that replacing the first row of G with the sum of the first two rows of G gives a matrix whose first row is divisible by $x+y$. In fact

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} G = \begin{bmatrix} x+y & x+y \\ y & x \end{bmatrix} = \begin{bmatrix} x+y & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ y & x \end{bmatrix}.$$

Therefore

$$\begin{aligned} G &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x+y & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ y & x \end{bmatrix} \\ &= \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x+y & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ y & x \end{bmatrix} \\ &= \begin{bmatrix} x+y & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ y & x \end{bmatrix} \\ &= T\,\bar{G}, \end{aligned}$$

which is the desired factorization.

In the above example it was easy to see by inspection an elementary row operation that could be performed on G so as to bring the first row to a vector multiple of $x+y$. However this is not always the case.

For instance, if we let $D = \mathbb{F}_2[x, y]$ and let

$$G = \begin{bmatrix} x\,y+1 & x+y & x+1 \\ 0 & x+y^2 & x\,y^2+x+1 \end{bmatrix},$$

then the full-size minors of G have a common factor of $x\,y+1$. But it is not clear how to perform row operations (or even if appropriate row operations can be performed) so that the first row of the resulting matrix has a factor of $x\,y+1$.

The following simple lemma follows immediately from the definition of a quotient ring.

Lemma 3.2.12 *Let D be a commutative ring. Suppose $G \in D^{k\times n}$ and $f \in D$. Also suppose $A \in D^{k\times k}$. Let $[G]$ and $[A]$ be the images of G and A under the canonical projection $D \longrightarrow D/(f)$ applied elementwise to G and A. So $[G] \in (D/(f))^{k\times n}$ and $[A] \in (D/(f))^{k\times k}$.*

The first row of $A\,G \in D^{k\times n}$ has every entry divisible by f if and only if the first row of $[A][G] \in (D/(f))^{k\times n}$ is the zero row.

Proof: Immediate since $[A][G] = [A\,G] \in (D/(f))^{k\times n}$. □

From this lemma it follows that if the full-size minors of $G \in D^{k\times n}$ have a common factor of $f \in D$, we may be able to determinantally factor G with respect to f by row reducing $[G] \in (D/(f))^{k\times n}$ until a row of zeros appears. Then the unimodular matrix U that gives this row of zeros may be multiplied on the left of G to give a row that is a vector multiple of f, and then Lemma 3.2.10 may be applied to obtain a determinantal factorization of G.

The next lemma gives the basis for a determinantal factorization technique in the situation that the ground ring is a polynomial ring in one variable over a PID.

Lemma 3.2.13 *Let P be a PID with Q the field of fractions of P. Let x be an indeterminate.*

Suppose $G = G(x) \in P[x]^{k\times n}$ $(k \le n)$, and all the full-size minors of G are divisible by the primitive first degree polynomial $f = f(x) = a\,x+b$ where a, $b \in P$ with $a \ne 0$ (to say the polynomial is primitive here means that a and b have no nonunit common factors in P). Then there exists a unimodular matrix $U \in P^{k\times k}$ such that the top row of $U\,G$ has all entries divisible by f.

Proof: As above we use the notation $[M]$ to represent the image in a quotient ring of a matrix M. By primitivity of f, we have that if $g \in P[x]$ and f divides g in $Q[x]$, then f divides g in $P[x]$ (see for instance [19, Lemma 16.18]). We make the easy observation that working in the quotient ring $Q[x]/(f)$ we have that $x = -\frac{b}{a}$.

Thus $[G(-\frac{b}{a})] = [G(x)] \in (Q[x]/(f))^{k\times n}$. Note that all denominators of $G(-\frac{b}{a})$ are powers of a, and so there is a nonnegative integer r such that $a^r\,G(-\frac{b}{a}) \in P[x]^{k\times n}$. We then have

$$[a^r\,G(-\frac{b}{a})] = [a^r\,G(x)] \in (Q[x]/(f))^{k\times n}.$$

But then these two matrices are also equal in $(P[x]/(f))^{k\times n}$ by primitivity of f.

Now all the full-size minors of $[G(-\frac{b}{a})] \in (Q[x]/(f))^{k\times n}$ are zero since all the full-size minors of $G(x)$ are divisible by $a\,x+b$. Therefore all the full-size minors of $[a^r\,G(-\frac{b}{a})] \in (P[x]/(f))^{k\times n}$ are zero. However all entries of $[a^r\,G(-\frac{b}{a})]$ are elements of P. Therefore all full-size minors of $a^r\,G(-\frac{b}{a})$ are elements of P and divisible by $a\,x+b$. This can only occur if all the full-size minors of $a^r\,G(-\frac{b}{a})$ are zero.

We therefore have a matrix over a PID, all of whose full-size minors are zero, and so there is a unimodular matrix $U \in P^{k\times k}$ such that $U\,a^r\,G(-\frac{b}{a})$ has its first row all zeros. (see for instance Jacobson [20, sect. 3.7]).

Then the matrix $[a^r\,U\,G(-\frac{b}{a})] \in (P[x]/(f))^{k\times n}$ has a row of zeros. But then the equal matrix $[a^r\,U\,G(x)] \in (P[x]/(f))^{k\times n}$ has a row of zeros, and so then does $[U\,G(x)] \in (P[x]/(f))^{k\times n}$. Hence by Lemma 3.2.12, $U\,G(x) \in (P[x])^{k\times n}$ has all entries of its first row divisible by $f(x)$, as desired. □

Based on the three lemmas of this subsection, we now give a determinantal factorization algorithm with respect to a primitive first degree common factor of the full-size minors of a matrix $G(x)$ over a polynomial ring in one indeterminate over a PID. We remark that this algorithm depends on being able to perform row reduction in the PID to find U in step 3 below. For more general PID's, U may have to be obtained by some other method. In the case that the ring is a Euclidean domain, step 3 may in fact be performed by row reduction.

Algorithm 3.2.14

Given: P a PID, a matrix $G(x) \in P[x]^{k\times n}$, and a primitive first degree common factor $f(x) = a\,x + b$ of the full-size minors of $G(x)$.

Goal: A determinantal factorization of $G(x)$ with respect to $a\,x + b$.

Step 1: Form the matrix $G(-\frac{b}{a})$.

Step 2: Clear denominators of $G(-\frac{b}{a})$ by multiplying every entry by the same nonzero entry of P. Call the resulting matrix $\tilde{G} \in P^{k\times n}$. The full-size minors of $\tilde{G}$ will all be zero.

Step 3: Form the augmented matrix $[I_k \mid \tilde{G}]$ and transform it by row operations to $[U \mid G_1]$ where the first row of G_1 is the zero row. Every entry of the first row of $U\,G(x)$ has a common factor of $a\,x + b$. (As remarked above, in an arbitrary PID, U may not be obtainable by row reduction.)

Step 4: Let $u = \det U$. Put

$$T = U^{-1}\left[\begin{array}{c|ccc} (a\,x+b)\,u & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & I_{k-1} & \\ 0 & & & \end{array}\right],$$

and

$$\bar{G} = \left[\begin{array}{c|ccc} \frac{u^{-1}}{a\,x+b} & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & I_{k-1} & \\ 0 & & & \end{array}\right] U\,G(x).$$

Note that $\bar{G}$ will have all entries in $P[x]$ since the denominators of $a\,x + b$ introduced by the first factor are cancelled by the multiples of $a\,x + b$ in the first row of $U\,G(x)$.

Then $G = T\,\bar{G}$ is the desired factorization.

We finish this subsection with two examples illustrating the use of the above algorithm.

Example 3.2.15 Here we will use the ring of integers, $\mathbb{Z}$, as our PID. Let

$$G(x) = \begin{bmatrix} 3\,x+1 & 4\,x+9 & 2\,x+5 \\ -x^2+x+52 & 2\,x^2-x+132 & x^2+x+68 \end{bmatrix} \in \mathbb{Z}[x]^{2\times 3}.$$

Each full-size minor of $G(x)$ is divisible by $f(x) = 5\,x - 8$. We now find a determinantal factorization of $G(x)$ with respect to $f(x)$ following the algorithm given above.

$$G(\frac{8}{5}) = \begin{bmatrix} \frac{29}{5} & \frac{77}{5} & \frac{41}{5} \\ \frac{1276}{25} & \frac{3388}{25} & \frac{1804}{25} \end{bmatrix}.$$

Clear denominators by multiplying by 25. We obtain

$$\tilde{G} = 25\,G(\frac{8}{5}) = \begin{bmatrix} 145 & 385 & 205 \\ 1276 & 3388 & 1804 \end{bmatrix}.$$

We then form the augmented matrix

$$\left[\begin{array}{cc|ccc} 1 & 0 & 145 & 385 & 205 \\ 0 & 1 & 1276 & 3388 & 1804 \end{array}\right] \overset{\text{row ops}}{\longrightarrow} \left[\begin{array}{cc|ccc} -44 & 5 & 0 & 0 & 0 \\ 9 & -1 & 29 & 77 & 41 \end{array}\right].$$

So $U = \begin{bmatrix} -44 & 5 \\ 9 & -1 \end{bmatrix}$. And

$$U\,G(x) = \begin{bmatrix} (5x-8)(-x+27) & (5x-8)(2x-33) & (5x-8)(x-15) \\ x^2+26x-43 & -2x^2+37x-51 & -x^2+17x-23 \end{bmatrix}.$$

Hence

$$\begin{aligned} & G(x) \\ &= U^{-1} \begin{bmatrix} -5x+8 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x+27 & -2x+33 & -x+15 \\ x^2+26x-43 & -2x^2+37x-51 & -x^2+17x-23 \end{bmatrix} \\ &= \begin{bmatrix} -5x+8 & 5 \\ -45x+72 & 44 \end{bmatrix} \begin{bmatrix} x+27 & -2x+33 & -x+15 \\ x^2+26x-43 & -2x^2+37x-51 & -x^2+17x-23 \end{bmatrix}, \end{aligned}$$

giving the desired factorization.

For our final example of this subsection we return to the example given in the discussion preceding Lemma 3.2.12.

Example 3.2.16 We work over the ring $\mathbb{F}_2[x,y]$. Let

$$G(x,y) = \begin{bmatrix} xy+1 & x+y & x+1 \\ 0 & x+y^2 & xy^2+x+1 \end{bmatrix}.$$

The full-size minors of $G(x,y)$ have a common factor of $xy+1$. This common factor is first degree in both x and y. So we may apply the above algorithm letting $y = \frac{1}{x}$ (using $\mathbb{F}_2[x][y]$ as our polynomial ring in one determinate, y over a PID, $\mathbb{F}_2[x]$)) or letting $x = \frac{1}{y}$ (using $\mathbb{F}_2[y][x]$). We will do the former.

$$G(x, \tfrac{1}{x}) = \begin{bmatrix} 0 & x+\frac{1}{x} & x+1 \\ 0 & x+\frac{1}{x^2} & \frac{1}{x}+x+1 \end{bmatrix}.$$

Then $x^2\,G(x,\frac{1}{x}) = \begin{bmatrix} 0 & x^3+x & x^3+x^2 \\ 0 & x^3+1 & x^3+x^2+x \end{bmatrix}$.

From this we obtain $U = \begin{bmatrix} x^2+x+1 & x^2+x \\ 1 & 1 \end{bmatrix}$.

And this leads to the determinantal factorization

$$G(x,y) = \begin{bmatrix} xy+1 & x^2+x \\ xy+1 & x^2+x+1 \end{bmatrix} \begin{bmatrix} x^2+x+1 & xy+x+y & x^2y+xy+x+1 \\ xy+1 & y^2+y & xy^2 \end{bmatrix}.$$

Remark 3.2.17 After we have given some more results in this chapter, we will mention another determinantal factorization technique in Section 3.5.

3.3 Results on Modules

Let A be a commutative ring. Let M be an A-submodule of the free A-module, A^n. (As usual we consider elements of A^n to be row vectors.)

Definition 3.3.1 An *image representation of* M is a matrix $G \in A^{l \times n}$ such that $M = \text{rowspace}(G) = A^l \cdot G$.

A *kernel representation of* M is a matrix $H \in A^{n \times j}$ (j a positive integer) such that $M = \ker(H) = \{v \in A^n \mid v \cdot H = 0\}$.

Observe that M as above has an image representation if and only if M is finitely generated. On the other hand, as the following example shows, a finitely generated module M need not have a kernel representation.

Example 3.3.2 Let M be the $\mathbb{Z}$-submodule of $\mathbb{Z}^2$ generated by $[\,2 \quad 4\,]$. M cannot have a kernel representation because $[\,1 \quad 2\,]$ is not in M, but it is in the kernel of any matrix for which $[\,2 \quad 4\,]$ is in the kernel.

Definition 3.3.3 The *orthogonal module* of M is denoted by $M^\perp$. It is given by $M^\perp := \{w \in A^n \mid wv^t = 0 \text{ for all } v \in M\}$.

We next put the orthogonal module in an algebraic context. To do this we adopt the following notation. Let A, M be as above. For $c \in A^n$ we define

$$\begin{array}{rccl} \varphi_c : & A^n & \longrightarrow & A \\ & v & \longmapsto & v\,c^t \ . \end{array}$$

We let $\bar{\varphi}_c : M \longrightarrow A$ be the restriction of φ_c to M.

With this notation, $M^{\perp}$ is the kernel of the map

$$\begin{array}{ccl} A^n & \longrightarrow & \hom_A(M, A) \\ c & \longmapsto & \bar{\varphi}_c \ . \end{array}$$

That is the sequence

$$0 \longrightarrow M^{\perp} \longrightarrow A^n \longrightarrow \hom_A(M, A)$$

is exact.

We have the following proposition. A version of this result for vector spaces may be found in Fuhrmann [15, p. 80].

Proposition 3.3.4 *A, M as above. $M^{\perp} \cong \hom_A(A^n/M, A)$.*

Proof: We use the notation given above. For $c \in M^{\perp}, M \subseteq \ker(\varphi_c)$. So there is an induced map

$$\begin{array}{rccl} \tilde{\varphi_c} : & A^n/M & \longrightarrow & A \\ & x + M & \longmapsto & x\,c^t \ . \end{array}$$

We then define a map

$$\begin{array}{rccl} \psi : & M^{\perp} & \longrightarrow & \hom_A(A^n/M, A) \\ & c & \longmapsto & \tilde{\varphi_c} \ . \end{array}$$

We claim that ψ is an isomorphism.

First we show that ψ is injective. Suppose $b \in \ker(\psi)$. So $\tilde{\varphi_b}(x + M) = 0$ for all $x \in A^n$. But this implies that $x\,b^t = 0$ for all $x \in A^n$. Hence $b = 0$. So $\ker(\psi)$ is trivial and ψ is injective.

To show surjectivity, let $\varphi \in \hom_A(A^n/M, A)$. Let $\pi : A^n \longrightarrow A^n/M$ be the projection map. So $\varphi\,\pi : A^n \longrightarrow A$. Note that $M \subseteq \ker(\varphi\,\pi)$. Now since $\varphi\,\pi \in \hom_A(A^n, A)$, there exists $c \in A^n$ such that $\varphi\,\pi = \varphi_c$. Also since $M \subseteq \ker(\varphi\,\pi) = \ker(\varphi_c)$, we have $c \in M^{\perp}$. Therefore $\tilde{\varphi_c}$ is defined. For $x \in A^n$ we have

$$\varphi(x + M) = \varphi\,\pi(x) = \varphi_c(x) = x\,c^t = \tilde{\varphi_c}(x + M).$$

Hence $\varphi = \tilde{\varphi_c} = \psi(c)$. So ψ is surjective. □

Next we give some standard properties of $M^{\perp}$.

Lemma 3.3.5 *Let M be a finitely generated A-submodule of A^n. Say $M = \mathrm{rowspace}(G)$ where $G \in A^{l \times n}$. Then $M^{\perp} = \ker(G^t)$.*

Proof: Let $v \in M^{\perp}$. So $vw^t = 0$ for all $w \in M$. In particular this holds if w is a row of G. Hence $v \cdot G^t = 0$. So $v \in \ker(G^t)$. Hence $M^{\perp} \subseteq \ker(G^t)$.

On the other hand, if $v \in \ker(G^t)$, then $v \cdot G^t = 0$. Now for any $w \in M$ we have $w = p \cdot G$ for some $p \in A^l$. Then $vw^t = v \cdot G^t \cdot p^t = 0$. So $v \in M^{\perp}$. We have $\ker(G^t) \subseteq M^{\perp}$. The result follows. □

The following proposition is standard and elementary.

Proposition 3.3.6 *Let M, N be A-submodules of A^n.*

i) $M^{\perp}$ is an A-submodule of A^n.

ii) $M \subseteq M^{\perp\perp}$.

iii) If $M \subseteq N$, then $N^{\perp} \subseteq M^{\perp}$.

iv) $M^{\perp} = M^{\perp\perp\perp}$.

Proposition 3.3.7 *Let D be a commutative noetherian ring. Let M be a D-submodule of D^n. The following conditions are equivalent.*

i) $M = M^{\perp\perp}$.

ii) $M = N^{\perp}$ *for some D-submodule, N, of D^n.*

iii) M *has a kernel representation.*

Proof: $i) \Rightarrow ii)$: Obvious–let $N = M^{\perp}$.
$ii) \Rightarrow iii)$: Assume $M = N^{\perp}$. Since D is noetherian, the module N is finitely generated. Let H be an image representation of N. Say $H \in D^{l\times n}$. By Lemma 3.3.5, we have $\ker(H^t) = N^{\perp} = M$. So M has a kernel representation.
$iii) \Rightarrow i)$: Assume M has a kernel representation, say $M = \ker(H^t)$ where H is a matrix with n columns. Let $N = \text{rowspace}(H)$. We have $M = \ker(H^t) = N^{\perp}$ (the last equality follows from Lemma 3.3.5). Therefore we have $M^{\perp\perp} = N^{\perp\perp\perp} = N^{\perp} = M$ as desired. $\square$

3.3.1 Minor Prime $\Leftrightarrow$ Kernel Representation

In this subsection we establish the following theorem.

Theorem 3.3.8 *Let D be a noetherian UFD. Let $G \in D^{k\times n}$. Suppose G has a nonzero $k \times k$ minor. Let $M = \text{rowspace}(G)$. M has a kernel representation if and only if G is minor prime.*

A different proof of this result, found independently, and using more advanced techniques of commutative algebra, is given by Wood, Rogers, and Owens in the preprint [54]. Their proof is stated only for rings of characteristic zero, although it appears that it could be generalized to arbitrary characteristic. A proof of this result for a Laurent polynomial ring in several indeterminates recently appeared in a paper of Fornasini and Valcher [10]. Also the dual version (for systems) of this theorem in the 2-D case may be found in Rocha [38, p.72, p.178]. The proof we give here, found independently, uses only elementary properties of matrices, and is more general, applying to any noetherian UFD.

We require several preliminary results.

Lemma 3.3.9 *Let D be a UFD. Let $b, v_1, v_2, ..., v_k \in D$ have no nonunit common factors. Suppose $c \in D$. If $b|(v_i c)$ for $1 \leq i \leq k$, then $b|c$.*

Proof: Suppose to the contrary that $b \nmid c$. Then there exists an irreducible element $p \in D$ and an integer $j \geq 1$ such that $p^j | b$ but $p^j \nmid c$. But then we must have $p | v_i$ for $1 \leq i \leq k$. Hence p is a nonunit common factor of $b, v_1, v_2, ..., v_k$, a contradiction. $\square$

Lemma 3.3.10 *Let D be a UFD and $G' \in D^{k\times k}$. Let $b \in D$, and $v, w \in D^k$ with $v = [\, v_1 \quad v_2 \quad \cdots \quad v_k \,]$, such that $b, v_1, v_2, ..., v_k$ have no nonunit common factors. Suppose $v \cdot G' = b \cdot w$. Then $b|\det(G')$.*

Proof: We have $v \cdot G' \cdot (\text{adj } G') = b \cdot w \cdot (\text{adj } G')$. Hence $\det(G') \cdot v = b \cdot w \cdot (\text{adj } G')$. Therefore $b | v_i \cdot \det(G')$ for $1 \leq i \leq k$. By Lemma 3.3.9 we have $b|\det(G')$ as desired. $\square$

Corollary 3.3.11 *Let D be a UFD, $G \in D^{k\times n}$ $(k \leq n)$. Let $b \in D$, $v \in D^k$ with $v = [\, v_1 \quad v_2 \quad \cdots \quad v_k \,]$, and $w \in D^n$. Suppose $b, v_1, v_2, ..., v_k$ have no nonunit common factors. If $v \cdot G = b \cdot w$, then b divides every $k \times k$ minor of G.*

Proof: Let G' be any $k \times k$ submatrix of G. Say G' consists of columns $i_1, ..., i_k$ of G where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$. Let w' be the $1 \times k$ submatrix $[\, w_{i_1} \quad \cdots \quad w_{i_k} \,]$. So $v \cdot G' = b \cdot w'$. So by Lemma 3.3.10, $b|\det(G')$. This holds for all $k \times k$ submatrices G' of G. $\square$

Proposition 3.3.12 (Minor prime $\Rightarrow$ kernel representation) *Let D be a Noetherian UFD. Let $G \in D^{k \times n}$ have a nonzero $k \times k$ minor. Suppose G is minor prime. Let $M =$ rowspace G. Then M has a kernel representation.*

Proof: By virtue of Propositions 3.3.6 and 3.3.7, we need only prove $M^{\perp\perp} \subseteq M$.

Let $Q(D)$ be the field of fractions of D. We will write $M_Q := \text{rowspace}_{Q(D)}(G)$ for the vector subspace of $Q(D)^n$ generated by the rows of G. Observe that M_Q has dimension k since G has a nonzero $k \times k$ minor.

Let $w \in M^{\perp\perp}$. Then $w \in {M_Q}^{\perp\perp}$ (here with regard to M_Q, the perp operator is applied with respect to the field $Q(D)$). Now ${M_Q}^{\perp\perp} = M_Q$ since this is the case of a vector space over a field where the equality is well known (for instance a simple dimension argument can be used along with Proposition 3.4.2 below).

So there is a vector $v = \begin{bmatrix} \frac{v_1}{b} & \cdots & \frac{v_k}{b} \end{bmatrix} \in Q(D)^k$, (where $v_i, b \in R, b \neq 0$), such that $v \cdot G = w$. Here we may assume that $v_1, ..., v_k, b$ have no nonunit common factors in R (else we may reduce all $\frac{v_i}{b}$ by cancelling the same nonunit common factor from b and from all the v_i's).

Then $\begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix} \cdot G = b \cdot w$. By Corollary 3.3.11, b divides every full-size minor of G. Since G is minor prime, this means that b is a unit in D. So $v \in D^k$, and consequently, $w = v \cdot G \in M$.

This shows that $M^{\perp\perp} \subseteq M$, and the proof is complete. $\square$

It is interesting to note that Proposition 3.3.12 fails if the condition of G being minor prime is replaced by G being left factor prime. We illustrate this point with the following important example.

Example 3.3.13 [cf. Proposition 3.1.7]

Let $G = \begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{bmatrix}$ over the ring $R = \mathbb{F}_2[z_1, z_2, z_3]$. As shown in Proposition 3.1.7, G is left factor prime. Letting $M =$ rowspace(G), we have that $M^{\perp\perp}$ contains precisely the elements w of R^3 that satisfy $w \cdot \begin{bmatrix} z_2 & z_3 & z_1 \end{bmatrix}^t = 0$. In particular, $\begin{bmatrix} z_3 & z_2 & 0 \end{bmatrix} \in M^{\perp\perp}$. However it is clear that $\begin{bmatrix} z_3 & z_2 & 0 \end{bmatrix} \notin M$ (we need to take a rational combination of the rows of G to produce this element). So here $M^{\perp\perp} \neq M$. Thus by Proposition 3.3.7, M does not have a kernel representation.

Proposition 3.3.12 establishes half of Theorem 3.3.8. The converse also requires some preliminary results.

Lemma 3.3.14 *Let D be a UFD. Let $G \in D^{k \times n}$ and $d \in D$. Suppose d divides every $k \times k$ minor of G but d does not divide every $(k-1) \times (k-1)$ minor of G. Then there exists $v \in D^k$ such that d divides vG but d does not divide v (here to say a ring element divides a vector means that the ring element divides each entry of the vector).*

Proof: Without loss of generality we may assume that d does not divide the upper left $(k-1) \times (k-1)$ minor of G. Let $\tilde{G}$ be the leftmost $k \times k$ submatrix of G. So d divides $\det(\tilde{G})$ but d does not divide every $(k-1) \times (k-1)$ minor of $\tilde{G}$.

Now consider $(\text{adj } \tilde{G}) \cdot G$. By Lemma 3.2.2, every nonzero entry of this matrix is (possibly with a sign change) a $k \times k$ minor of G (for consider multiplying adj $\tilde{G}$ by each column of G to obtain the product $(\text{adj } \tilde{G}) \cdot G$). Hence every entry of $(\text{adj } \tilde{G}) \cdot G$ is divisible by d.

Now some entry of adj $\tilde{G}$ is not divisible by d (for each $(k-1) \times (k-1)$ minor of $\tilde{G}$ is up to sign an entry of adj $\tilde{G}$, and we know that at least one of these minors is not divisible by d). Let v be a row of adj $\tilde{G}$ that contains an entry not divisible by d. Then d does not divide v. But vG is a row of $(\text{adj } \tilde{G}) \cdot G$. So d divides vG. Hence v is the desired vector. $\square$

Corollary 3.3.15 *Let D and G be as in Lemma 3.3.14. If $d \in D$ is a nonunit and d divides every $k \times k$ minor of G, then there exists $v \in R^k$ such that d divides vG but d does not divide v.*

Proof: There is a least integer q such that d divides every $q \times q$ minor of G. Clearly $1 \leq q \leq k$.

Case 1: $q = 1$. Then d divides every entry of G. We take $v = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \end{bmatrix}$.

Case 2: $q > 1$. By definition of q there is a $(q-1) \times (q-1)$ minor of G that is not divisible by d. Without loss of generality we may assume that the upper left $(q-1) \times (q-1)$ minor of G is not divisible by d.

Let $\tilde{G} \in D^{q \times n}$ be the submatrix of G formed from the first q rows of G. Apply Lemma 3.3.14 to $\tilde{G}$, thus obtaining $\tilde{v} \in D^q$ such that d divides $\tilde{v}\tilde{G}$ but d does not divide $\tilde{v}$. Now let $v \in D^k$ be given by $v = [\ \tilde{v} \mid 0 \ \ \dots \ \ 0\ ]$.

Then $vG = \tilde{v}\tilde{G}$. So d divides vG but not v. We have found (constructively) the desired v. □

Corollary 3.3.16 *Let D, G, and d be as in Corollary 3.3.15. Suppose also that G has a nonzero $k \times k$ minor. Then there exists $w \in D^n$ such that $d \cdot w \in \operatorname{rowspace}(G)$ but $w \notin \operatorname{rowspace}(G)$.*

Proof: Choose $v \in D^k$ such that d divides vG but d does not divide v. Then $vG = d \cdot w$ for some $w \in D^n$. So $d \cdot w \in \operatorname{rowspace}(G)$.

However we claim that $w \notin \operatorname{rowspace}(G)$. To see this, suppose to the contrary that $w \in \operatorname{rowspace}(G)$. Then $w = v_1 G$ for some $v_1 \in D^k$. Then we have $d \cdot v_1 G = d \cdot w = vG$. Therefore $(d \cdot v_1 - v)G = 0$. But since G has a nonzero $k \times k$ minor, its left kernel is trivial. So $d \cdot v_1 - v = 0$. Then $d \cdot v_1 = v$, a contradiction since d does not divide v. Hence the claim and the corollary hold. □

Corollary 3.3.17 (Kernel representation ⇒ minor prime) *Let D be a UFD. Let $G \in D^{k \times n}$ be of rank k. If G is not minor prime, then $\operatorname{rowspace}(G)$ does not have a kernel representation.*

Proof: Since G is not minor prime there is a nonunit $d \in D$ such that d is a common factor of every full size minor of G.

Choose $w \in D^n$ such that $d \cdot w \in \operatorname{rowspace}(G)$ but $w \notin \operatorname{rowspace}(G)$.

Now consider any matrix H such that $\operatorname{rowspace}(G) \subseteq \ker(H)$. We have $(d \cdot w)H = 0$. But then $wH = 0$. So $w \in \ker(H)$. Hence $\operatorname{rowspace}(G) \neq \ker(H)$. Therefore $\operatorname{rowspace}(G)$ does not have a kernel representation. □

Proposition 3.3.12 and Corollary 3.3.17 together establish Theorem 3.3.8.

3.4 Applications to m-D Codes

In this section we will consider how the material of the previous sections of this chapter applies to m-D convolutional codes as developed in Chapter 2. Recall that $R = \mathbb{F}[z_1, \dots, z_m]$ is the polynomial ring in m indeterminates over a finite field.

Definition 3.4.1 For an m-D convolutional code $\mathcal{C}$, we refer to the module $\mathcal{C}^\perp$ as the *orthogonal code* of $\mathcal{C}$. If $\mathcal{C}$ has a kernel representation, say $\mathcal{C} = \ker(H^t)$, we say H is a *parity check matrix* for $\mathcal{C}$. Also in the case where $\mathcal{C}$ has a kernel representation, we call the orthogonal code $\mathcal{C}^\perp$ the *dual code* of $\mathcal{C}$.

We have the following important relation between the rank of a code and the rank of its orthogonal code.

Proposition 3.4.2 *Let $\mathcal{C}$ be a rate $\frac{k}{n}$ m-D convolutional code. Then the rate of $\mathcal{C}^\perp$ is $\frac{n-k}{n}$. That is, $\operatorname{rank}(\mathcal{C}) + \operatorname{rank}(\mathcal{C}^\perp) = n$.*

Proof: Let $G \in R^{l \times n}$ be a generator matrix for $\mathcal{C}$. Recall from Definition 2.2.1 that $\operatorname{rank}(\mathcal{C})$ is the dimension of the vector subspace of $Q(R)^n$ generated by the rows of G. By virtue of Lemma 3.3.5 we have that

$$\begin{aligned} \operatorname{rank}(\mathcal{C}) + \operatorname{rank}(\mathcal{C}^\perp) &= \dim_{Q(R)}(\operatorname{rowspace}(G)) + \dim_{Q(R)}(\ker(G^t)) \\ &= \dim_{Q(R)}(\operatorname{rowspace}(G^t)) + \dim_{Q(R)}(\ker(G^t)) . \end{aligned}$$

Now consider the map $Q(R)^n \xrightarrow{\cdot G^t} Q(R)^l$. By the dimension theorem we have that

$$\dim(\operatorname{image}(\cdot G^t)) + \dim(\ker(\cdot G^t)) = n ,$$

which when compared with the previous equation gives the desired result. □

The following corollary is immediate.

Corollary 3.4.3 *Let $\mathcal{C}$ be a m-D convolutional code. Then* $\mathrm{rank}(\mathcal{C}^{\perp\perp}) = \mathrm{rank}(\mathcal{C})$.

A parity check matrix, H, for the code $\mathcal{C}$ is desirable in that it allows us to check whether $v \in R^n$ is a codeword: in fact v is a codeword if and only if $v \cdot H^t = 0$. We know from the previous section (see Theorem 3.3.8) that if $\mathcal{C}$ is free with a minor prime encoder, then $\mathcal{C}$ has a parity check matrix.

In the cases of block codes and 1-D convolutional codes, freeness of the dual code is not an issue (all codes are free in these cases). However in the m-D case where $m \geq 2$, if $\mathcal{C}$ has a dual code, the dual code need not be free, even if $\mathcal{C}$ itself is free. Next we give an algebraic condition for $\mathcal{C}^\perp$ to be free.

Lemma 3.4.4 *Let $G \in R^{l \times n}$. Let $\mathcal{C} = \mathcal{C}(G)$ be the code generated by G. Let $N = \mathrm{rowspace}(G^t)$. $\mathcal{C}^\perp$ is free if and only if the sequence*

$$R^n \xrightarrow{\cdot G^t} N \longrightarrow 0$$

can be extended to a short exact sequence of the form

$$0 \longrightarrow R^p \longrightarrow R^n \xrightarrow{\cdot G^t} N \longrightarrow 0 \ .$$

Proof: First we assume that $\mathcal{C}^\perp$ is free. Now by Lemma 3.3.5, $\mathcal{C}^\perp = \ker(G^t)$. So $\ker(G^t) \cong R^p$ for some $p \geq 0$. Let $\tilde{\varphi} : R^p \longrightarrow \ker(G^t)$ be an R-isomorphism. Let $\varphi : R^p \longrightarrow R^n$ be given by $(v)\varphi = (v)\tilde{\varphi}$. Then

$$0 \longrightarrow R^p \xrightarrow{\varphi} R^n \xrightarrow{\cdot G^t} N \longrightarrow 0$$

is a short exact sequence.

Conversely suppose we have a short exact sequence

$$0 \longrightarrow R^p \xrightarrow{\psi} R^n \xrightarrow{\cdot G^t} N \longrightarrow 0 \ .$$

Then $\mathrm{image}(\psi) = \ker(\cdot G^t) = \mathcal{C}^\perp$. But ψ considered as a map $R^p \longrightarrow \mathrm{image}(\psi)$ is an isomorphism. So $\mathrm{image}(\psi)$ is free. So then also is $\mathcal{C}^\perp$. □

Proposition 3.4.5 *If $\mathcal{C}$ is any 2-D convolutional code, then $\mathcal{C}^\perp$ is free.*

Proof: In this situation, $R = \mathbb{F}[z_1, z_2]$. Let $G \in R^{l \times n}$ and let $\mathcal{C} = \mathcal{C}(G)$ be the code generated by G. It is well known that any finitely generated module over $\mathbb{F}[z_1, ..., z_m]$ has a free resolution of length m or less (a proof may be found in Adams and Loustaunau [1, p. 197]). In the case $m = 2$ this amounts to saying that any finitely generated module N can be placed in a short exact sequence of the form

$$0 \longrightarrow R^p \longrightarrow R^q \longrightarrow N \longrightarrow 0 \qquad (p, q \in \mathbb{N}).$$

We apply this to $\mathrm{rowspace}(G^t)$ obtaining a short exact sequence

$$0 \longrightarrow R^p \longrightarrow R^q \longrightarrow \mathrm{rowspace}(G^t) \longrightarrow 0 \ .$$

We also have the epimorphism $R^n \xrightarrow{\cdot G^t} \mathrm{rowspace}(G^t)$. We note that $\mathcal{C}^\perp$ is the kernel of this epimorphism. Thus we have another short exact sequence:

$$0 \longrightarrow \mathcal{C}^\perp \longrightarrow R^n \longrightarrow \mathrm{rowspace}(G^t) \longrightarrow 0 \,.$$

We may apply Schanuel's Lemma (see for instance Northcott [32, Thm. 3, p. 26]) to our two exact sequences. This allows us to conclude that

$$R^p \oplus R^n \cong R^q \oplus \mathcal{C}^\perp.$$

Hence $\mathcal{C}^\perp$ is a projective module. Therefore $\mathcal{C}^\perp$ is free by the Quillen-Suslin theorem (see for instance Lam [23] or Lang [24, pp. 846-850]).

□

Corollary 3.4.6 *If the* 2*-D convolutional code* $\mathcal{C}$ *has a kernel representation, then* $\mathcal{C}$ *is free.*

Proof: To say $\mathcal{C}$ has a kernel representation means $\mathcal{C}$ is the orthogonal code of some 2-D convolutional code, and the result then follows immediately from Proposition 3.4.5. □

Corollary 3.4.7 *Every* 2*-D convolutional code* $\mathcal{C}$ *is contained in a free convolutional code of the same rank.*

Proof: We have $\mathcal{C} \subseteq \mathcal{C}^{\perp\perp}$. And by Proposition 3.4.5, $\mathcal{C}^{\perp\perp}$ is free. The result now follows from Corollary 3.4.3. □

The following example shows that Corollaries 3.4.6 and 3.4.7 may fail if $m \geq 3$.

Example 3.4.8 In this example let $R = \mathbb{F}_2[z_1, z_2, z_3]$. Let $H = \begin{bmatrix} z_2 & z_3 & z_1 \end{bmatrix}$. Let $\mathcal{D}$ be the code generated by H. Let $\mathcal{C} = \ker(H^t)$. So $\mathcal{C}$ has a kernel representation. In fact $\mathcal{C} = \mathcal{D}^\perp$. Clearly $\mathcal{D}$ is free of rank 1. So by Proposition 3.4.2 $\text{rank}(\mathcal{C}) = 2$.

We will show that $\mathcal{C}$ does not have a 2×3 generator matrix, which will allow us to conclude that $\mathcal{C}$ is not free.

Suppose to the contrary that $G_1 \in R^{2\times 3}$ is a generator matrix for $\mathcal{C}$. Now observe that

$$\begin{bmatrix} z_1 & 0 & z_2 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & z_1 & z_3 \end{bmatrix} \in \mathcal{C}\,.$$

Therefore there exists $T \in R^{2\times 2}$ such that

$$T \cdot G_1 = \begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{bmatrix}.$$

However this last matrix is the one found in Proposition 3.1.7, and from that proposition we know it is left factor prime. Therefore T is unimodular. Hence by Corollary 2.2.7 we have that $\begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{bmatrix}$ is also an encoder for $\mathcal{C}$.

Observe that $\begin{bmatrix} z_3 & z_2 & 0 \end{bmatrix} \in \mathcal{C}$, but $\begin{bmatrix} z_3 & z_2 & 0 \end{bmatrix}$ is not in the code generated by $\begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{bmatrix}$. This is a contradiction. So we must conclude that $\mathcal{D}^\perp = \mathcal{C}$ is not free. So Corollary 3.4.6 can fail for $m \geq 3$.

We continue this example by showing that $\mathcal{C}$ is not contained in a free code of rank 2. We suppose to the contrary that $\mathcal{C}$ is contained in a rank 2 free code, $\tilde{\mathcal{C}}$. Let $\tilde{G} \in R^{2\times 3}$ be a generator matrix for $\tilde{\mathcal{C}}$. Now it is not hard to show that

$$G = \begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \\ z_3 & z_2 & 0 \end{bmatrix}$$

is a generator matrix for $\mathcal{C}$. Since $\mathcal{C} \subseteq \tilde{\mathcal{C}}$ we have that there is a matrix $A \in R^{3\times 2}$ such that $A \cdot \tilde{G} = G$. Let T be the 2×2 submatrix formed from the first two rows of A. We have

$$T \cdot \tilde{G} = \begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{bmatrix}.$$

But as before, such T must be unimodular, from which it follows that $\begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{bmatrix}$ is an encoder for $\mathcal{C}$. But this contradicts the fact that $\mathcal{C}$ is not free. We conclude that no such $\tilde{\mathcal{C}}$ can exist, showing the failure of Corollary 3.4.7.

Next we take up the issue of recovering a message word from a codeword. The idea is simple: if v is the message word and G is the encoder, then $w = v \cdot G$ is the corresponding codeword. To recover v we must invert the action of G on v.

Definition 3.4.9 Let $G \in R^{k\times n}$ be of rank k. So G is an encoder. An *inverter* for G is a rational matrix $P \in \mathbb{F}(z_1, ..., z_m)^{n\times k}$ such that $GP = I_k$, the $k \times k$ identity matrix.

An inverter P of G is a *polynomial inverter* if $P \in R^{n\times k}$ (i.e., P is a polynomial matrix).

Remark 3.4.10

i) If one is transmitting information over a noiseless channel (i.e. a channel over which no transmission errors occur), a message $v \in R^k$ is encoded as $w = v \cdot G$. Then w is decoded by computing $w \cdot P = (v \cdot G)P = v \cdot I_k = v$.

ii) Assuming that $k < n$, a polynomial inverter for $G \in R^{k\times n}$ (if one exists) is not unique. If P is any polynomial inverter, another polynomial inverter may be obtained by adding to any column of P the transpose of any element of $\mathcal{C}(G)^\perp$

iii) A necessary and sufficient condition for $G \in R^{k\times n}$ of rank k to have a polynomial inverter, P, is for the exact sequence

$$0 \longrightarrow R^k \xrightarrow{\cdot G} R^n \longrightarrow R^n/\mathcal{C} \longrightarrow 0$$

to be split exact. ($P : R^n \longrightarrow R^k$ splits the map $\cdot G$.)

The proof of the next proposition relies on the Cauchy-Binet formula which we recall here. A proof of this formula may be found in the article by Zeng [58].

Lemma 3.4.11 (The Cauchy-Binet formula) *Let A and B be matrices of size $k \times n$ and $n \times k$ respectively ($k \leq n$) over any commutative ring. Let $\Lambda = \{i = (i_1, ..., i_k) : 1 \leq i_1 < i_2 < \ldots < i_k \leq n\}$. For $i \in \Lambda$ let A_i be the $k \times k$ submatrix of A formed from columns $i_1, ..., i_k$ of A, and let B_i be the $k \times k$ submatrix of B formed from rows $i_1, ..., i_k$ of B.*

Then

$$\det(A \cdot B) = \sum_{i\in\Lambda} \det(A_i)\det(B_i)\,.$$

The following proposition can essentially be found in the paper [57] by Youla and Pickel.

Proposition 3.4.12 *Let $G \in R^{k\times n}$. G has a polynomial inverter if and only if G is zero prime. In this case there exist matrices $P \in R^{n\times k}$, $\hat{G} \in R^{(n-k)\times n}$, and $\hat{P} \in R^{n\times(n-k)}$ such that the following five equations hold:*

$$G \cdot P = I_k \qquad G \cdot \hat{P} = 0_{k\times(n-k)}$$
$$\hat{G} \cdot P = 0_{(n-k)\times k} \qquad \hat{G} \cdot \hat{P} = I_{n-k}$$
$$P \cdot G + \hat{P} \cdot \hat{G} = I_n\ .$$

Proof: Suppose G has a polynomial inverter P. So $G \cdot P = I_k$. So $\det(G \cdot P) = 1$. By the Cauchy-Binet formula (Lemma 3.4.11), $\det(G \cdot P)$ is a polynomial combination of the full-size minors of G. So 1 is in the ideal J generated by the full-size minors of G.

Conversely suppose that G is zero prime. Then by the Quillen-Suslin theorem (see [57, 23] or [24, pp. 846-850]), there exists a matrix $\hat{G}$ such that $\begin{bmatrix} G \\ \hat{G} \end{bmatrix}$ is unimodular. Then there exists $\tilde{P} \in R^{n\times n}$ with $\tilde{P} = [P\ \hat{P}]$ where $P \in R^{n\times k}$ and $\hat{P} \in R^{n\times(n-k)}$ such that

$$\begin{bmatrix} G \\ \hat{G} \end{bmatrix} \begin{bmatrix} P & \hat{P} \end{bmatrix} = I_n\ .$$

Then $G \cdot P = I_k$ so G has a polynomial inverter. The five equations in the statement of the proposition follow at once from

$$\begin{bmatrix} G \\ \hat{G} \end{bmatrix} \begin{bmatrix} P & \hat{P} \end{bmatrix} = I_n = \begin{bmatrix} P & \hat{P} \end{bmatrix} \begin{bmatrix} G \\ \hat{G} \end{bmatrix}.$$

□

Recall from the discussion preceding Lemma 3.4.4 and from Example 3.4.8 that a free code that has a kernel representation (and hence a dual code) may have a nonfree dual code. The following corollary shows that the situation is better in the case of a code that has a zero prime generator matrix.

Corollary 3.4.13 *Suppose $G \in R^{k\times n}$ is zero prime. Let $\mathcal{C}$ be the code $\mathcal{C}(G)$ generated by G. Since G is minor prime, it has a dual code. This dual code is free.*

Proof: We use the notation of the previous proposition. Consider the sequence

$$0 \longrightarrow R^{n-k} \xrightarrow{\cdot \hat{P}^t} R^n \xrightarrow{\cdot G^t} \text{image}(\cdot G^t) \longrightarrow 0.$$

We claim that this sequence is exact. First note that since $\hat{G} \cdot \hat{P} = I_{n-k}$, we have that $\hat{P}$ is of rank $n-k$, and then also $\hat{P}^t$ is of rank $n-k$. Then since $\hat{P}^t \in R^{(n-k)\times n}$, it follows that the sequence is exact at R^{n-k}. Clearly the sequence is exact at image$(\cdot G^t)$. It remains to show exactness at R^n. We have

$$\hat{P}^t \cdot G^t = (G \cdot \hat{P})^t = 0^t = 0,$$

and so image$(\cdot \hat{P}^t) \subseteq$ ker$(\cdot G^t)$. Now suppose that $w \in$ ker$(\cdot G^t)$. Then we have

$$\begin{aligned} w &= w\ I_n = w(P \cdot G + \hat{P} \cdot \hat{G})^t = w(G^t \cdot P^t + \hat{G}^t \cdot \hat{P}^t) \\ &= w\ G^t \cdot P^t + w\ \hat{G}^t \cdot \hat{P}^t = 0 + w\ \hat{G}^t \cdot \hat{P}^t. \end{aligned}$$

Therefore $w \in$ image$(\cdot \hat{P}^t)$. So the sequence is exact at R^n, establishing the claim.

Then by Lemma 3.4.4, the dual code, $\mathcal{C}^\perp$ is free. □

3.5 On Factor Primeness

It has been noted in the literature (see for instance, Zerz [59], Fornasini and Valcher [10], or Peterson, Rosenthal, and Weiner [34]) that for a matrix $G \in R^{k\times n}$ of rank k, the notions of zero primeness and minor primeness of G have simple algebraic geometric equivalents.

In particular, zero primeness of G is equivalent to the condition that the full-size minors of G define the empty variety in $\bar{\mathbb{F}}^m$ where $\bar{\mathbb{F}}$ is the algebraic closure of $\mathbb{F}$. And minor primeness of G is equivalent to the condition that the variety in $\bar{\mathbb{F}}^m$ defined by the full-size minors of G does not contain any hypersurface.

It also has been observed in [59, 10] that left factor primeness does not seem to have a simple algebraic or algebraic geometric characterization. In [10] it is stated that currently an algorithmic test for left factor primeness is not available for $m \geq 3$. Of course for $m \leq 2$, left factor primeness is equivalent to minor primeness which can be tested using the Euclidean Algorithm for $m = 1$ or Gröbner basis techniques for $m \geq 2$.

The results from the previous sections of this chapter can be used to give some information on the factor primeness of a matrix $G \in R^{k\times n}$ of rank k. In fact we will show that given a matrix G as above we can always draw one of the following three conclusions: (i) G is definitely left factor prime; (ii) G is definitely not left factor prime, but G has a right factor of the same size as G that is minor prime (and hence this factor is also left factor prime); or (iii) G or a right factor of G of the same size as G is left factor prime but not minor prime.

We begin with a lemma.

Lemma 3.5.1 *Suppose $A \in R^{k\times n}$ and $B \in R^{l\times n}$ are both of rank k. Suppose also that* rowspace$(A) \subseteq$ rowspace(B). *Then* (rowspace$(A))^\perp$ = (rowspace$(B))^\perp$.

Proof: Letting $Q(R)$ be the field of fractions of R, we have that as vector spaces, rowspace$_{Q(R)}(A) \subseteq$ rowspace$_{Q(R)}(B)$. However both are of rank k, and so (being in the situation of vector spaces) rowspace$_{Q(R)}(A)$ = rowspace$_{Q(R)}(B)$.

Therefore there is a matrix $M \in Q(R)^{l\times k}$ such that $B = M\,A$.

Now suppose $w \in$ (rowspace$_R(A))^\perp$. We have $wA^t = 0$. But then $wB^t = w(M\,A)^t = wA^tM^t = 0$. So $w \in$ (rowspace$_R(B))^\perp$. Hence (rowspace$_R(A))^\perp \subseteq$ (rowspace$_R(B))^\perp$. The reverse inclusion holds by Proposition 3.3.6 (iii). □

We now show how one of the three conclusions preceding the lemma may always be drawn.

From Proposition 3.1.2, if G is minor prime, then G is left factor prime, so in this case conclusion (i) above may be drawn.

Of course the more interesting case is when G is not minor prime, and then G may or may not be left factor prime (recall Proposition 3.1.7 for an example of a matrix that is left factor prime but not minor prime).

In this situation we find a matrix $H \in R^{(n-k)\times n}$ of rank $n-k$ such that $G\,H^t = 0$ (one way to find such a matrix H is to use Gaussian elimination on G over the field of rational functions to find the kernel of G over this field. Then clearing denominators of a basis matrix of this kernel gives the desired matrix H). Note that $\mathcal{C}(H) \subseteq \mathcal{C}(G)^\perp$ and is of the same rank as $\mathcal{C}(G)^\perp$. Hence from Lemma 3.5.1 it follows that $\mathcal{C}(H)^\perp = \mathcal{C}(G)^{\perp\perp}$.

We then compute a minimal free resolution of the module generated by (the rows of) H^t. (See for instance Adams and Loustaunau [1, Ch. 3]). If the free resolution has the form

$$0 \longrightarrow R^p \longrightarrow R^n \xrightarrow{\cdot H^t} \text{rowspace}(H^t) \longrightarrow 0$$

for some $p \geq 0$, then by Lemma 3.4.4 $\mathcal{C}(H)^\perp$ is free. Hence we can find a generator matrix $\bar{G} \in R^{k\times n}$ of $\mathcal{C}(H)^\perp = \mathcal{C}(G)^{\perp\perp}$. Since $\mathcal{C}(\bar{G})$ has a kernel representation (namely $\mathcal{C}(\bar{G}) = \ker(H^t)$), we have by Theorem 3.3.8 that $\bar{G}$ is minor prime. Then since $\mathcal{C}(G) \subseteq \mathcal{C}(G)^{\perp\perp} = \mathcal{C}(\bar{G})$, we have that there exists a matrix T in $R^{k\times k}$ such that $G = T\,\bar{G}$. Since G is not minor prime and $\bar{G}$ is minor prime, T cannot be unimodular. So in this case we draw conclusion (ii) that G is definitely not left factor prime.

Next suppose H is as above, but that the minimal free resolution of rowspace(H^t) is not a short exact sequence. In this case $\mathcal{C}(H)^\perp = \mathcal{C}(G)^{\perp\perp}$ is not free. Hence $\mathcal{C}(G)^{\perp\perp}$ does not have a minor prime generator matrix. We claim that G has a right factor of size $k \times n$ that is left factor prime but not minor prime (this includes the possibility that G itself is the stated right factor of G). To see that the claim is correct, suppose to the contrary that G has no such right factor. Then G itself is not left factor prime, and so a factorization $G = T\,G_1$ where $T \in R^{k\times k}$, $G_1 \in R^{k\times n}$, and $\det(T)$ is of positive degree may be obtained. Moreover, we may choose the factorization so that $\det(T)$ is of largest possible degree. Then G_1 is left factor prime (otherwise the maximality of degree $\det(T)$ would be violated). But G_1 is not minor prime (if it were $\mathcal{C}(G_1)$ would generate $\mathcal{C}(G)^{\perp\perp}$ which would then be a free module, a contradiction). So the claim holds. And in this situation, conclusion (iii) may be drawn.

Remark 3.5.2 For a specific matrix G, the calculations needed to draw the above conclusions can be performed using available computer software packages such as CoCoA[1]. This package will compute the minimal free resolution of a module, and can be used to compute $\mathcal{C}(G)^\perp$ for a polynomial matrix G.

The bulk of the calculations for the following example were done using CoCoA.

Example 3.5.3 We work in three dimensions over the binary field $\mathbb{F}_2$. We use $x, y,$ and z as our indeterminates instead of $z_1, z_2,$ and z_3.

Let

$$G = \begin{bmatrix} x & y & z & 0 & y \\ y & x & z & x & 0 \\ x & y & 0 & y & z \\ x & x & 0 & 0 & x+z \end{bmatrix}.$$

The full-size minors of G have a greatest common divisor of $(x+y)z$. So G is not minor prime; it may or may not be left factor prime. Using CoCoA we find that $\mathcal{C}(G)^\perp$ is generated by

$$H = [\, x\,z + y\,z \quad x^2 + y\,z \quad x^2 + x\,y + y^2 \quad x^2 + x\,z + y\,z \quad x^2\,].$$

A minimal length free resolution of rowspace(H^t) is given by

$$0 \longrightarrow R^2 \xrightarrow{M_1} R^6 \xrightarrow{M_2} R^5 \xrightarrow{H^t} \text{rowspace}(H^t) \longrightarrow 0,$$

where

$$M_1 = \begin{bmatrix} 0 & x & 0 & y & 1 & 0 \\ 0 & 0 & y^2 & 0 & x+y & z \end{bmatrix}$$

[1]CoCoA was developed by A. Giovini and G. Niesi, Department of Mathematics, University of Genova, via L.B. Alberti 4, 16132, Genova ITALY. It is a software package for doing algebraic geometric and Gröbner basis calculations. Assuming that this address is still current, CoCoA is available for free, and may be requested from the developers. CoCoA may also be downloaded from http://cocoa.dima.unige.it/ on the Internet.

and

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & z & y & y+z \\ 0 & x+y & z & 0 & x+y+z \\ 0 & y & z & x & x+y \\ 0 & y^2 & xz+yz & 0 & y^2+xz \\ 0 & 0 & x^2 & 0 & x^2+xy+y^2 \end{bmatrix}.$$

Because this resolution is longer than a short exact sequence, we draw conclusion (iii): some 4×5 right factor of G is left factor prime but not minor prime.

Indeed using the fact that M_2 is a generator matrix for $(\mathcal{C}(G))^{\perp\perp}$, it is not too hard to find the factorization

$$G = \begin{bmatrix} x & 0 & 0 & 1 \\ y & 1 & 1 & 1 \\ x & 1 & 0 & 1 \\ x & 0 & 1 & 1 \end{bmatrix} \tilde{M_2},$$

where $\tilde{M_2}$ is the submatrix formed by the top four rows of M_2. The left factor in this factorization of G has determinant $x+y$, and the greatest common divisor of the full-size minors of $\tilde{M_2}$ is z. Then $\tilde{M_2}$ must be the right factor of G that is left factor prime (for if it were not we could extract a non-unimodular square left factor from $\tilde{M_2}$, but the determinant of this factor would then have to be z–since z is the only nonunit common factor of the full-size minors of $\tilde{M_2}$. The remaining right factor would be minor prime, but this would contradict conclusion (iii)).

Remark 3.5.4 Considerations from this section give another method for obtaining determinantal factorizations over R (see also Section 3.2).

This method is based on the fact that rowspace$(G) \subseteq$ (rowspace$(G))^{\perp\perp}$. Therefore a generator matrix $\bar{G} \in R^{l \times n}$ of (rowspace$(G))^{\perp\perp}$ is a right factor of G. If (rowspace$(G))^{\perp\perp}$ is free, then $\bar{G}$ may be taken minor prime in $R^{k \times n}$, giving a right factor of G that is the same size as G and minor prime.

In any case, if H is as above, taking a resolution of rowspace(H^t) will give a generating set for (rowspace$(G))^{\perp\perp}$, namely the matrix M in this resolution:

$$\cdots \longrightarrow R^l \xrightarrow{M} R^n \xrightarrow{H^t} \text{rowspace}(H^t).$$

Then even if $l > k$ it may be possible to find a $k \times n$ submatrix, $\tilde{M}$ of M such that rowspace$(G) \subseteq$ rowspace$(\tilde{M})$, and then $\tilde{M}$ is a right factor of G of the same size as G. This is exactly the situation that occurred in Example 3.5.3.

CHAPTER 4

UNIT MEMORY CODES

In this chapter we begin by reviewing monomial orders for a set of indeterminates. Then we introduce unit memory codes. We consider a special type of unit memory code, deriving a lower bound on the distance for codes of this type. Then we look at a decoding technique for this type of unit memory code.

Throughout this chapter, $R = \mathbb{F}[z_1, ..., z_m]$ is a polynomial ring in m indeterminates over the finite field $\mathbb{F} = \mathbb{F}_q$. We also use multi-index notation (see Section 2.4). For a multi-index $\alpha = (\alpha_1, ..., \alpha_m) \in \mathbb{N}^m$, we use the notation $|\alpha| := \alpha_1 + ... + \alpha_m$. For a monomial $\mathbf{z}^\alpha$ we call $|\alpha|$ the *total degree* of the monomial.

Also any polynomial will have coefficients that are scalars, vectors, or matrices over $\mathbb{F}$. When necessary we will specify the source of the coefficients, but for the sake of generality we will not always do so. In any case if we discuss a product $f \cdot g$ of polynomials, we assume that this product is defined (i.e., that the coefficients of f may be multiplied by the coefficients of g) and that a nonzero coefficient of f times a nonzero coefficient of g is nonzero. So for instance in a product $f \cdot g$, f may have coefficients in $\mathbb{F}^k$ and g may have coefficients in $\mathbb{F}^{k \times n}$ of full row rank, and then $f \cdot g$ has coefficients in $\mathbb{F}^n$. The requirement on nonzero products of coefficients is exactly what we will need for our code construction in Section 4.2.

4.1 Monomial Orders

Most of the results in this section are standard. References for this material include the books by Adams and Loustaunau [1], Cox, Little, and O'Shea [3], and Eisenbud [4].

Definition 4.1.1 A *monomial order* on the indeterminates $z_1, ..., z_m$ is a total order, $\prec$, on the set of monomials formed with these indeterminates such that

i) $1 \prec \mathbf{z}^\alpha$ for all $\alpha \in \mathbb{N}^m$ with $\alpha \neq (0, 0, ..., 0)$, and

ii) if $\alpha, \beta, \gamma \in \mathbb{N}^m$ with $\mathbf{z}^\alpha \prec \mathbf{z}^\beta$, then $\mathbf{z}^\alpha \mathbf{z}^\gamma \prec \mathbf{z}^\beta \mathbf{z}^\gamma$.

Regarding such a total order, $\prec$, we will sometimes write $\alpha \prec \beta$ for α, $\beta \in \mathbb{N}^m$ to mean $\mathbf{z}^\alpha \prec \mathbf{z}^\beta$. We use $\preceq, \succ$, and $\succeq$ in the usual ways.

Example 4.1.2 One important type of monomial order is *lexicographic* monomial order. We choose any ordering of linear monomials, say $z_m \prec z_{m-1} \prec \ldots \prec z_1$. Then for $\alpha = (\alpha_1, ..., \alpha_m)$, $\beta = (\beta_1, ..., \beta_m) \in \mathbb{N}^m$, $\mathbf{z}^\alpha \prec \mathbf{z}^\beta$ if $\alpha_i < \beta_i$ where i is the index of the greatest linear monomial (with respect to the chosen linear monomial ordering) at which α and β disagree.

For instance $z_1^3 z_2 z_3^5 \prec z_1^3 z_2^2$.

Given a different ordering of the linear monomials, lexicographic ordering would be determined by comparing first the exponents on the greatest linear monomials, then (if there was a tie on the first) on the second greatest linear monomial, and so on. Hence there are $m!$ distinct lexicographic orders on m indeterminates, one for each ordering of $z_1, ..., z_m$.

Example 4.1.3 Another monomial order we will have occasion to use is *degree lexicographic* order. As in the previous example, an ordering for linear monomials is chosen. Then monomials are first ordered by total degree, and then in case of a tie, lexicographic order is used.

So for instance if $z_m \prec z_{m-1} \prec \ldots \prec z_1$ is the chosen linear monomial order, then with degree lexicographic order we have $z_1^2 z_2 \prec z_2^4 \prec z_1 z_2^3$. (Contrast this with lexicographic order, where these three monomials would be ordered by $z_2^4 \prec z_1 z_2^3 \prec z_1^2 z_2$.)

An important difference between lexicographic and degree lexicographic orders is that with respect to a degree lexicographic order, each monomial has only finitely many predecessors. Hence in considering monomials in sequence with degree lexicographic ordering, every monomial appears in a finite position in the sequence. Also of interest is the fact that in considering monomials in a degree lexicographic order, all monomials of a certain total degree are considered before any monomials of a higher degree.

Remark 4.1.4 The previous two examples show that for $m \geq 2$, there are at least $2(m!)$ different monomial orders on $z_1, \ldots, z_m$. In fact for $m \geq 2$ there are uncountably many different monomial orders on $z_1, \ldots, z_m$.

A monomial order allows us to order the terms of a polynomial, and in particular it allows us to identify a *leading term* and a *trailing term* (which are respectively the terms with the greatest and the least monomial part). We will denote the leading and trailing terms of a polynomial, f, by $\mathrm{LT}(f)$ and $\mathrm{TT}(f)$. We observe that different monomial orders may give rise to different leading terms and different trailing terms. We will call a term of the polynomial f an *extreme term* if it is the leading or trailing term of f with respect to some monomial order.

We now state some simple properties of monomial orders.

Lemma 4.1.5 *Let $\prec$ be a monomial order for $z_1, \ldots, z_m$, and let $\alpha, \beta, \gamma, \delta \in \mathbb{N}^m$. If $\mathbf{z}^\alpha \preceq \mathbf{z}^\beta$ and $\mathbf{z}^\gamma \preceq \mathbf{z}^\delta$, then $\mathbf{z}^\alpha \mathbf{z}^\gamma \preceq \mathbf{z}^\beta \mathbf{z}^\delta$. Equality holds only if $\alpha = \beta$ and $\gamma = \delta$.*

Corollary 4.1.6 *Suppose f and g are polynomials for which $f \cdot g$ is defined. For a fixed monomial order, we have $\mathrm{LT}(f \cdot g) = \mathrm{LT}(f)\mathrm{LT}(g)$, and $\mathrm{TT}(f \cdot g) = \mathrm{TT}(f)\mathrm{TT}(g)$. (As noted in the beginning of the chapter, we are assuming that a nonzero coefficient of f times a nonzero coefficient of g is nonzero.)*

Definition 4.1.7 Let f and g be polynomials for which $f \cdot g$ is defined. A *term product* of $f \cdot g$ is the product of a term of f and a term of g.

The following proposition, which follows easily from Lemma 4.1.5 and Corollary 4.1.6, gives an essential property of extreme terms of a product of polynomials.

Proposition 4.1.8 *Let f and g be polynomials for which the product $f \cdot g$ is defined. Any extreme term of $f \cdot g$ is formed by a single term product. That is only one term product of $f \cdot g$ has the monomial part of the given extreme term. In particular, while a general nonzero term product of $f \cdot g$ may be cancelled by some other term product(s), a term product of the form $\mathrm{LT}(f)\mathrm{LT}(g)$ or $\mathrm{TT}(f)\mathrm{TT}(g)$ (where these terms are with respect to a fixed monomial order) cannot be cancelled by any other term products.*

Example 4.1.9 Over $\mathbb{F}_2[x, y]$ let $f = 1 + x + y + y^2$ and $g = x + x^2 + xy + y^2$. There are only four possible orders for the monomials involved in f and g. We give them here along with the corresponding leading and trailing terms of f and g.

order	$\mathrm{LT}(f)$	$\mathrm{LT}(g)$	$\mathrm{TT}(f)$	$\mathrm{TT}(g)$
$1 \prec x \prec y \prec x^2 \prec xy \prec y^2$	y^2	y^2	1	x
$1 \prec y \prec x \prec y^2 \prec xy \prec x^2$	y^2	x^2	1	x
$1 \prec x \prec x^2 \prec y \prec xy \prec y^2$	y^2	y^2	1	x
$1 \prec y \prec y^2 \prec x \prec xy \prec x^2$	x	x^2	1	y^2

So the extreme terms of $f \cdot g$ are $y^4,\ x^2 y^2,\ x^3,\ x,$ and y^2.

We may apply Proposition 4.1.8 to this example: There must be at least five terms in the product $f \cdot g$ since there are five extreme terms.

4.2 Unit Memory Codes

The following definition may be regarded as a generalization to the m-D case of the concept of unit memory encoders, described in papers by Thommesen and Justesen [46] and Lee [25].

Definition 4.2.1 A *unit memory code* is an m-D convolutional code $\mathcal{C}$ that has a generator matrix with all polynomials of first degree. A *unit memory encoder* is a matrix $G \in R^{k\times n}$ of rank k, all the entries of which are first degree.

A unit memory code has a generator matrix of the form $G = G_0 + z_1G_1 + \cdots + z_mG_m$ where $G_i \in \mathbb{F}^{l\times n}$ $(0 \leq i \leq m)$ is a matrix with entries in the ground field $\mathbb{F}$. We have had good results considering a special type of unit memory codes. We give the motivation for and description of this below.

Piret gives a 1-D code construction in [36] in which he forms a parity check matrix as $H_0 + zH_1$ where H_0, H_1 are $(n-k)\times n$ parity check matrices of Reed-Solomon codes. (See for instance MacWilliams and Sloane [28, Chapter 10] for a description of Reed-Solomon codes). The code construction we suggest here was inspired by the construction of Piret, but it is different in several ways. First we define the generator matrix directly; second we do not restrict ourselves to using Reed-Solomon codes; and third we generalize our construction to the m-D case.

If G is a unit memory encoder we may write $G = G_0 + z_1\ G_1 + \cdots + z_m\ G_m$ where each $G_i \in \mathbb{F}^{k\times n}$ is a scalar matrix. We get interesting results if we require each G_i to be the generator matrix of a rate $\frac{k}{n}$ block code (i.e., each G_i is of rank k). We impose this requirement throughout the rest of this chapter. The construction follows.

Let $C_0, C_1, ..., C_m$ be rate $\frac{k}{n}$ block codes with generator matrices $G_0, G_1, ..., G_m \in \mathbb{F}^{k\times n}$. Let $\mathrm{dist}(C_i) = \delta_i, i = 0, 1, ..., m$. Let

$$G = G_0 + z_1G_1 + \cdots + z_mG_m \in R^{k\times n}.$$

Let $\mathcal{C}$ be the m-D code generated by G.

First we prove that with this construction that G is a unit memory encoder.

Lemma 4.2.2 *Let G and G_i $(0 \leq i \leq m)$ be as in the above construction. G is of rank k and consequently G generates a free rate $\frac{k}{n}$ m-D convolutional code.*

Proof: We must show that G has a nonzero $k \times k$ minor. Now G_0 is the generator of a rate $\frac{k}{n}$ block code and hence has a nonzero $k\times k$ minor. Then writing $G = G(z_1, ..., z_m)$, we have $G(0, ..., 0) = G_0$, and so the corresponding $k \times k$ minor of G must be nonzero. $\square$

Next we give an important lower bound on the distance of the code $\mathcal{C}$ in terms of its constituent block codes.

Proposition 4.2.3 *Using the notation immediately preceding Lemma 4.2.2, we have* $\mathrm{dist}(G) \geq \delta_0 + \delta_1 + \cdots + \delta_m$.

Proof: Let v be any nonzero element of R^k. We must show that $\mathrm{wt}(v\cdot G) \geq \delta_0 + \delta_1 + \cdots + \delta_m$.

Choose a lexicographic monomial ordering of R that gives z_1 as the greatest linear monomial. Let $v_1\mathbf{z}^{\alpha_1}$ be the leading term of v with respect to this order (here $v_1 \in \mathbb{F}^k$). Then since z_1G_1 is the leading term of G with respect to this ordering, we have, by Proposition 4.1.8, that $v_1G_1\mathbf{z}^{\alpha_1}z_1$ is an extreme term of the polynomial product vG . The weight of this term is at least δ_1 because v_1G_1 is a nonzero codeword of C_1, and we know that $\mathrm{dist}(C_1) = \delta_1$.

We can repeat this argument m times, the j^{th} time choosing a lexicographic monomial ordering that gives z_j as the greatest linear monomial. In this manner we find m distinct terms of vG with weights $\delta_1, \ldots, \delta_m$.

Next using any lexicographic monomial ordering, we apply the same argument to a trailing term. Because the trailing term of G is G_0 we find that there is a term of vG (distinct from the m terms constructed above) whose coefficient is a nonzero codeword of C_0 and hence whose weight is at least δ_0.

Therefore $\mathrm{wt}(vG) \geq \delta_0 + \delta_1 + \cdots + \delta_m$. And so also $\mathrm{dist}(\mathcal{C}) \geq \delta_0 + \delta_1 + \cdots + \delta_m$. $\square$

In the following example we show that the inequality given in the statement of Proposition 4.2.3 may be strict.

Example 4.2.4 In this example we construct a 1-D code over the binary field $\mathbb{F}_2$. Since we are in the 1-D case, we use z as our indeterminate rather than z_1. Let

$$G_0 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \text{ and } G_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Then $\delta_0 = \delta_1 = 1$. Also $G = G_0 + zG_1 = \begin{bmatrix} 1+z & 1 & 1+z \\ 1 & 1+z & 0 \end{bmatrix}$. Let $\mathcal{C}$ be the code generated by G.

We claim that $\text{dist}(\mathcal{C}) = 3$. First observe that for any $v = \begin{bmatrix} f & g \end{bmatrix} \in R^2$ we have that $vG = \begin{bmatrix} (1+z)f + g & f + (1+z)g & (1+z)f \end{bmatrix}$. Now if, contrary to our claim, $\text{dist}(\mathcal{C}) < 3$, then for some nonzero $v = \begin{bmatrix} f & g \end{bmatrix} \in R^2$, $\text{wt}(vG) \le 2$. Hence at least one of the three components of vG must be 0.

Case 1: $(1+z)f + g = 0$. Then $g = (1+z)f$, and $vG = \begin{bmatrix} 0 & z^2 f & (1+z)f \end{bmatrix}$. Since $f \neq 0$ (else $v = 0$), vG has at least three nonzero terms. That is $\text{wt}(vG) \ge 3$.

Case 2: $f + (1+z)g = 0$. Then $f = (1+z)g$, and $vG = \begin{bmatrix} z^2 g & 0 & (1+z^2)g \end{bmatrix}$. Then since $g \neq 0$ we have that $\text{wt}(vG) \ge 3$.

Case 3: $(1+z)f = 0$. Then $f = 0$. So $vG = \begin{bmatrix} g & (1+z)g & 0 \end{bmatrix}$. Again $\text{wt}(vG) \ge 3$.

So in all cases, under the assumption that $\text{wt}(vG) < 3$, we obtain the contradiction that $\text{wt}(vG) \ge 3$. So we must conclude that $\text{dist}(\mathcal{C}) \ge 3$. Finally since $\text{wt}(\begin{bmatrix} 0 & 1 \end{bmatrix} G) = 3$, we have that $\text{dist}(\mathcal{C}) = 3$, establishing the claim.

So in this example, $\text{dist}(\mathcal{C}) > \delta_0 + \delta_1$.

Remark 4.2.5 It is interesting to note that in this construction, $\text{dist}(\mathcal{C})$ depends not only on the block codes $C_0, C_1, \ldots,$
C_m, but also on the generator matrices $G_0, G_1, \ldots, G_m$ chosen for these codes. In the above example, let us replace G_0 by $\tilde{G_0} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, which generates the same block code as G_0. We then get as generator matrix for the convolutional code

$$\tilde{G} = \tilde{G_0} + zG_1 = \begin{bmatrix} 1+z & 1 & 1+z \\ 0 & z & 1 \end{bmatrix}.$$

The corresponding code, $\tilde{\mathcal{C}}$, has distance 2. (For instance $\text{wt}(\begin{bmatrix} 0 & 1 \end{bmatrix} \tilde{G}) = 2$.)

We note here that in the above construction, the convolutional code $\mathcal{C}$ need not have a parity check matrix. For instance, if $G_0 = G_1 = \ldots = G_m$ then $1 + z_1 + z_2 + \ldots + z_m$ is a common factor of every entry of $G = G_0 + z_1 G_1 + \ldots + z_m G_m$. So G is not minor prime. Therefore by Theorem 3.3.8, $\mathcal{C}$ does not have a parity check matrix.

Recall here the block code bound from Section 2.5. After Example 2.5.4 we promised to give for any r an example of a rate $\frac{2}{3}$ code using r different monomials that has distance $2r$, thus meeting the block code bound.

Example 4.2.6 We choose $m = r - 1$. Let

$$G = \begin{bmatrix} 1 + z_1 + \cdots + z_{r-1} & 1 + z_1 + \cdots + z_{r-1} & 0 \\ 0 & 1 + z_1 + \cdots + z_{r-1} & 1 + z_1 + \cdots + z_{r-1} \end{bmatrix}.$$

Observe that G is a unit memory encoder and may be written as $G = G_0 + z_1 G_1 + \cdots + z_{r-1} G_{r-1}$ where $G_i = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$ for $0 \le i \le r-1$. Since each G_i has distance 2, we have by Proposition 4.2.3 that $\text{dist}(\mathcal{C}(G)) \ge 2r$. Thus $\mathcal{C}(G)$, which actually has distance equal to $2r$, meets the block code bound by Example 2.5.4.

The final example of this section serves to show that in the multidimensional case a rate $\frac{1}{1}$ convolutional code can correct errors (this does not occur in the 1-D case where at best one error may be detected).

Example 4.2.7 Let $G = [1 + z_1 + ... + z_m] \in R^{1\times 1}$. Since the rate $\frac{1}{1}$ block code with generator matrix [1] has distance 1, it follows that the code generated by G has distance $m+1$ and hence can correct up to $\lfloor \frac{m}{2} \rfloor$ errors. In fact the decoding algorithm to be developed in the next section applies to this example.

4.3 Decoding Unit Memory Codes

Our goal for this section is to develop an algorithm for decoding unit memory codes. Before getting into specific details, we give a general framework for decoding.

Suppose $G \in R^{k\times n}$ generates an m-D code, $\mathcal{C}$ of rate $\frac{k}{n}$. Let $\text{dist}(\mathcal{C}) \geq \delta$ and let $t = \left\lfloor \frac{\delta-1}{2} \right\rfloor$. So $\mathcal{C}$ corrects at least t errors by Lemma 2.4.5.

A message word $v \in R^k$ is encoded as $w = vG \in R^n$. The codeword w is transmitted over a noisy channel, and $y = w + e \in R^n$ is received. Here $e \in R^n$ is the error introduced in transmission. If we assume that $\text{wt}(e) \leq t$ then w is the unique codeword nearest to y.

We now give a procedure for attempting to recover v from y. This procedure is very similar to the Euclidean Algorithm for determining a quotient and remainder in dividing an integer a by a positive integer b. Here we think of "dividing" y by G and obtaining the "quotient" v and the "remainder" e. This procedure is not an algorithm because in step 3, selection of Δv is unspecified. Also as presented, the procedure may not halt. However with appropriate details, which we will give in this section, this procedure may be developed into an algorithm for decoding unit memory codes.

We use a right arrow ($\longrightarrow$) as an assignment operator: first the expression on the left of the arrow is evaluated, and then the result is stored in the location on the right side of the arrow. In this procedure, $\tilde{v} \in R^k$, and $\tilde{e} \in R^n$. $\tilde{v}$ is an estimate of v. In each pass through the loop of the procedure, at step 3 Δv is determined if possible. In step 4, $\tilde{v}$ is incremented by the amount Δv and becomes the new estimate for v. Also $\tilde{e}$ represents what the transmission error would be if $\tilde{v} = v$. If at some point $\text{wt}(\tilde{e}) \leq t$, then $\text{dist}(\tilde{v}G, y) \leq t$, and so we must have $\tilde{v} = v$ (since there is a unique codeword within a distance t of the received word y), and then correct decoding has occurred. As this section progresses we will specify how Δv may be selected so as to ensure the decoding of unit memory codes.

Procedure 4.3.1

Step 1: $0 \longrightarrow \tilde{v}, \quad y \longrightarrow \tilde{e}.$ (Here $0 \in R^k$).

Step 2: If $\text{wt}(\tilde{e}) \leq t$, go to step 6.

Step 3: Select $\Delta v \in R^k$, if possible. If Δv cannot be selected, go to step 7.

Step 4: $\tilde{v} + \Delta v \longrightarrow \tilde{v}, \quad \tilde{e} - (\Delta v)G \longrightarrow \tilde{e}.$

Step 5: Go to step 2.

Step 6: Stop. Successful decoding. $v = \tilde{v}$.

Step 7: Failure.

We next show that if step 6 is reached in this procedure, then y has been correctly decoded.

Lemma 4.3.2 *At the end of every step of the above procedure,*

$$\tilde{v}G + \tilde{e} = y.$$

Proof: The result clearly holds after step 1. The only other step where values of $\tilde{v}$ and $\tilde{e}$ change is step 4. Let $\tilde{v}_{\text{old}}$ and $\tilde{e}_{\text{old}}$ be the values of $\tilde{v}$ and $\tilde{e}$ upon entering step 4 on some pass through the loop formed by steps 2 to 5. Let $\tilde{v}_{\text{new}}$ and $\tilde{e}_{\text{new}}$ be the values of $\tilde{v}$ and $\tilde{e}$ upon leaving step 4. We have $\tilde{v}_{\text{new}} = \tilde{v}_{\text{old}} + \Delta v$, and $\tilde{e}_{\text{new}} = \tilde{e}_{\text{old}} - (\Delta v)G$. Therefore

$$\tilde{v}_{\text{new}}G + \tilde{e}_{\text{new}} = (\tilde{v}_{\text{old}} + \Delta v)G + (\tilde{e}_{\text{old}} - (\Delta v)G) = \tilde{v}_{\text{old}}G + \tilde{e}_{\text{old}}.$$

So the value of $\tilde{v}G + \tilde{e}$ does not change in passing through step 4. Since $\tilde{v}G + \tilde{e} = y$ the first time through step 4 (because of the initialization step 1), we have that $\tilde{v}G + \tilde{e} = y$ always. $\square$

Proposition 4.3.3 *Assume that in an application of Procedure 4.3.1,* $\mathrm{wt}(e) \le t$*. If step 6 is reached in the procedure, then* $\tilde{v} = v$*, and* $\tilde{e} = e$*, and so correct decoding of* y *is achieved.*

Proof: In order to reach step 6, we must pass through step 2 with $\mathrm{wt}(\tilde{e}) \le t$. By the previous lemma we have $y - \tilde{v}G = \tilde{e}$. So

$$\mathrm{dist}(y, \tilde{v}G) = \mathrm{wt}(y - \tilde{v}G) = \mathrm{wt}(\tilde{e}) \le t.$$

But we know that vG is the unique codeword within distance t of y. Hence $\tilde{v}G = vG$. Then $\tilde{v} = v$. The rest of the proposition follows immediately. □

For the remainder of this section the following notation, most of which is carried over from the previous section, will be used. $G = G_0 + z_1G_1 + \cdots + z_mG_m$ where $G_j \in \mathbb{F}^{k\times n}$ is a generator matrix of a rate $\frac{k}{n}$ block code, C_j, of distance δ_j. Let $\mathcal{C} = \mathcal{C}(G) \subseteq R^n$ be the unit memory code generated by G. For $0 \le j \le m$, let $t_j = \left\lfloor \frac{\delta_j - 1}{2} \right\rfloor$. So by Lemma 2.4.5, t_j is the maximum number of errors that can be corrected by the block code C_j. Let $\delta = \delta_0 + \delta_1 + \cdots + \delta_m$ and let $t = \lfloor \frac{\delta-1}{2} \rfloor$. By Proposition 4.2.3, $\mathrm{dist}(\mathcal{C}) \ge \delta$, and so $\mathcal{C}$ corrects at least up to t errors by Lemma 2.4.5. If $p \in R^n$ or R^k and $\alpha \in \mathbb{N}^m$, then $p_\alpha \in \mathbb{F}^n$ or $\mathbb{F}^k$ is the $\mathbf{z}^\alpha$-coefficient of p.

Let $v \in R^k$, $w = vG \in \mathcal{C}$, and $y = w + e \in R^n$ be a message word, the corresponding codeword, and the received word (as in the beginning of this section).

For the decoding algorithm for $\mathcal{C}$ that we will develop, we make three assumptions which will be in effect for the rest of this section.

Assumption 1: For $0 \le j \le m$, the receiver can perform C_j-decoding of up to t_j errors. That is, given a vector $b \in \mathbb{F}^n$ that has a distance at most t_j from some codeword of C_j (which must be the nearest codeword to b by Lemma 2.4.5), then the receiver can actually find this C_j-codeword nearest b (presumably in an efficient manner).

Assumption 2: $\mathrm{wt}(e) \le t$. That is, code $\mathcal{C}$ can correct the number of errors that occurred in sending w and receiving y.

Assumption 3: The receiver knows when transmission of a codeword is complete.

Remark 4.3.4 Regarding this last assumption, the receiver need not know exactly when transmission is complete, but must be able to determine an upper bound, $d-1 \ge 0$, on the total degree of v based on the transmission received. Using this upper bound on the total degree of v, we will treat y as if it were of total degree d (in fact y will be of total degree at most d).

We now give an algorithm that attempts to decode y using only C_0-decoding. The algorithm either successfully decodes y, or it indicates failure. In either case the algorithm halts after a finite number of steps.

We assume that some degree lexicographic order, $\prec$, has been fixed (see Example 4.1.3). As in Procedure 4.3.1 we use the right arrow ($\longrightarrow$) as the assignment operator. Throughout the algorithm, α is used as a multi-index in $\mathbb{N}^m$. $\mathrm{INC}(\cdot)$ is the incrementation operator on multi-indices in $\mathbb{N}^m$ with respect to the order $\prec$. So $\mathrm{INC}(\alpha)$ denotes the multi-index immediately succeeding α. The symbol $-\infty$ is a special symbol whose only property is that $\mathrm{INC}(-\infty) = (0, 0, \ldots, 0) \in \mathbb{N}^m$.

We note that this algorithm follows the pattern of Procedure 4.3.1.

Algorithm 4.3.5

Input: $y \in R^n$ of total degree at most d. $y = w + e = vG + e$ for some $v \in R^k$, and $w, e \in R^n$, with $\mathrm{wt}(e) \le t$ (but v, w, e are unknown at the time of input).

Step 1: $0 \longrightarrow \tilde{v}$, $\quad y \longrightarrow \tilde{e}$, $\quad -\infty \longrightarrow \alpha$.

Step 2: If $\mathrm{wt}(\tilde{e}) \le t$, go to step 10.

Step 3: $\mathrm{INC}(\alpha) \longrightarrow \alpha$.

Step 4: Try to C_0-decode $\tilde{e}_\alpha$. If unsuccessful, go to step 9. If successful, obtain $x \in F^k$ such that $\mathrm{dist}(xG_0, \tilde{e}_\alpha) \le t_0$.

Step 5: $x\,\mathbf{z}^{\alpha} \longrightarrow \Delta v$.

Step 6: $\tilde{v} + \Delta v \longrightarrow \tilde{v}, \qquad \tilde{e} - (\Delta v)G \longrightarrow \tilde{e}$.

Step 7: If $\mathrm{wt}(\tilde{e}) \leq t$, go to step 10.

Step 8: If $|\mathrm{INC}(\alpha)| < d$, go to step 3.

Step 9: Failure. There exists $\alpha^* \preceq \alpha$ such that $\tilde{v}_{\alpha^*} \neq v_{\alpha^*}$. Stop.

Step 10: Successful decoding has occurred. $v = \tilde{v}$. Stop

The following remark gives an informal description of the preceding algorithm.

Remark 4.3.6 Step 1 is an initialization step. Step 2 checks whether $v = 0$.

Steps 3 through 8 form the main loop. α is the multi-index of the least undecoded term of $y - \tilde{v}G = \tilde{e}$ (i.e., for multi-indices $\beta \prec \alpha$, $\tilde{v}_\beta$ has been determined). In step 4, we attempt to C_0-decode $\tilde{e}_\alpha$. If we fail in the attempt, then the algorithm indicates failure and halts. (By Assumption 2, failure here occurs only if $\tilde{e}_\alpha$ is further than t_0 distant from any codeword of C_0.) If we succeed in C_0-decoding $\tilde{e}_\alpha$, we get an estimate x of v_α. We make appropriate changes to $\tilde{v}$ and $\tilde{e}$ in step 6. Step 7 checks to see whether successful decoding has occurred. Step 8 checks to see if there are more multi-indices at which to apply C_0-decoding.

We also note that since a degree lexicographic ordering is used, the algorithm goes through all $\alpha \in \mathbb{N}^m$ with $|\alpha| < d$.

Observe that because this algorithm is an instance of Procedure 4.3.1, we have that Lemma 4.3.2 holds for this algorithm. That is, at every step, $\tilde{v}G + \tilde{e} = y$. The next two results show that Algorithm 4.3.5 does what it is supposed to do.

Lemma 4.3.7 *Let $\beta^* \in \mathbb{N}^m$ be a fixed multi-index with $|\beta^*| < d$. Suppose for all $\beta \prec \beta^*$, in the pass through the loop of the algorithm with $\alpha = \beta$, at step 4 a value of x is obtained and that $x = v_\beta$. Then in the pass through the loop with $\alpha = \beta^*$, at step 4, we recover x with $x = v_{\beta^*}$ if and only if* $\mathrm{wt}(e_{\beta^*}) \leq t_0$.

Proof: When we begin step 4 with $\alpha = \beta^*$, we have

$$\begin{aligned} \tilde{e} &= y - \tilde{v}G &&= y - \sum_{\alpha \prec \beta^*} \tilde{v}_\alpha G \mathbf{z}^\alpha \\ &= y - \sum_{\alpha \prec \beta^*} v_\alpha G \mathbf{z}^\alpha &&= e + vG - \sum_{\alpha \prec \beta^*} v_\alpha G \mathbf{z}^\alpha \\ &= e + \sum_{\alpha \succeq \beta^*} v_\alpha G \mathbf{z}^\alpha. \end{aligned}$$

Note that the least term (with respect to $\prec$) of the summation in the final line is $v_{\beta^*} G_0 z^{\beta^*}$. So $\tilde{e}_{\beta^*} = e_{\beta^*} + v_{\beta^*} G_0$. Hence in applying C_0-decoding to $\tilde{e}_{\beta^*}$, we recover v_{β^*} if and only if $\mathrm{wt}(e_{\beta^*}) \leq t_0$. □

Proposition 4.3.8 *Algorithm 4.3.5 decodes y correctly (i.e., recovers v) if and only if for all $\beta \in \mathbb{N}^m$ with $|\beta| < d$, $dist(y_\beta, w_\beta) \leq t_0$, or equivalently, $wt(e_\beta) \leq t_0$.*

Also if this algorithm fails to decode y correctly, it halts in a finite number of steps and indicates failure.

Proof: The algorithm recovers v correctly if and only if for all β with $|\beta| < d$, during the pass through the loop with $\alpha = \beta$ we obtain $x = v_\beta$ in step 4. By the previous lemma this occurs if and only if $\mathrm{wt}(e_\beta) \leq t_0$ for all β with $|\beta| < d$.

For the second part of the proposition, note that if correct decoding is not achieved, step 10 cannot be reached, so success cannot be declared. On the other hand, the loop formed by steps 3 through 8 can be executed only finitely many times (since there are only finitely many $\alpha \in \mathbb{N}^m$ with $|\alpha| < d$). So eventually step 8 will be reached with $|\mathrm{INC}(\alpha)| = d$. At this point (if not sooner due to failure at some iteration of step 4), control passes to step 9, and failure is indicated. □

We will need the following lemma as we develop the general decoding algorithm.

Lemma 4.3.9
(i) For $0 \le j \le m$,

$$t - t_0 - \cdots - t_j - (j+1) \le \left\lfloor \frac{\delta_{j+1} + \cdots + \delta_m - 1}{2} \right\rfloor .$$

(ii) $t < (t_0 + 1) + \cdots + (t_m + 1)$.

Proof: Since $t_i = \lfloor \frac{\delta_i - 1}{2} \rfloor$, we have

$$\delta_i = \begin{cases} 2t_i + 1 & \text{if } \delta_i \text{ is odd} \\ 2t_i + 2 & \text{if } \delta_i \text{ is even.} \end{cases}$$

So $\delta_i \le 2t_i + 2$. Hence for $0 \le j \le m$ we have

$$\begin{aligned} t &= \lfloor \tfrac{\delta_0 + \cdots + \delta_m - 1}{2} \rfloor \\ &= \left\lfloor \tfrac{\delta_0 + \cdots + \delta_j}{2} + \tfrac{\delta_{j+1} + \cdots + \delta_m - 1}{2} \right\rfloor \\ &\le \left\lfloor \tfrac{(2t_0+2) + \cdots + (2t_j+2)}{2} + \tfrac{\delta_{j+1} + \cdots + \delta_m - 1}{2} \right\rfloor \\ &= t_0 + \cdots + t_j + (j+1) + \left\lfloor \tfrac{\delta_{j+1} + \cdots + \delta_m - 1}{2} \right\rfloor . \end{aligned}$$

(i) follows immediately.

For (ii) we apply (i) with $j = m$. We obtain

$$t \le t_0 + \cdots + t_m + (m+1) + \left\lfloor \frac{-1}{2} \right\rfloor < t_0 + \cdots + t_m + (m+1),$$

which is equivalent to the desired inequality. □

Remark 4.3.10 We have that

$$\sum_\alpha \mathrm{wt}(e_\alpha) = \mathrm{wt}(e) \le t.$$

Hence (ii) of the preceding lemma says that we cannot find $m+1$ distinct terms of e with weights respectively greater than $t_0, \ldots, t_m$, for if we could we would have $\mathrm{wt}(e) > t$, which we know is not the case.

Next we give an algorithm for decoding in the 1-D case. The ability to decode in the 1-D case will serve as a basis for an induction showing how to decode in the general m-D case.

For the 1-D case we use z instead of z_1 for our indeterminate. We have $G = G_0 + zG_1$. We will index terms of our polynomials with $i \in \mathbb{N}$ (instead of with α which we reserve for higher dimensional multi-indices).

We have received $y = vG + e$ where $\mathrm{wt}(e) \le t = \lfloor \frac{\delta - 1}{2} \rfloor$. Suppose that v is of total degree at most $d - 1 \ge 0$, so y is of total degree at most $d \ge 1$. We have

$$\begin{aligned} v &= v_0 + v_1 z + v_2 z^2 + \cdots + v_{d-1} z^{d-1}, \\ w = vG &= v_0 G_0 + (v_0 G_1 + v_1 G_0) z + (v_1 G_1 + v_2 G_0) z^2 + \cdots \\ &\qquad + (v_{d-2} G_1 + v_{d-1} G_0) z^{d-1} + (v_{d-1} G_1) z^d, \\ y = w + e &= y_0 + y_1 z + y_2 z^2 + \cdots + y_d z^d. \end{aligned}$$

Of course the only information available to us as we begin decoding is the matrix G and the received word y.

Algorithm 4.3.11

Input: $y \in R^n$ as above.

Step 1: Apply Algorithm 4.3.5 to try to decode y using only C_0-decoding. If successful decoding occurs, go to step 8.

Step 2: $d-1 \longrightarrow i$.

Step 3: C_1 decode $\tilde{e}_{i+1} = (v_i - \tilde{v}_i)G_1 + e_{i+1}$. $v_i - \tilde{v}_i$ will be correctly recovered. $(v_i - \tilde{v}_i)z^i \longrightarrow \Delta v$. (See Lemma 4.3.12 below for an explanation and justification of this step).

Step 4: $\tilde{v} + \Delta v \longrightarrow \tilde{v}, \qquad \tilde{e} - (\Delta v)G \longrightarrow \tilde{e}$.

Step 5: If $\mathrm{wt}(\tilde{e}) \leq t$, go to step 8.

Step 6: $i-1 \longrightarrow i$.

Step 7: Go to step 3.

Step 8: Stop. Successful decoding has occurred. $v = \tilde{v}$ and $e = \tilde{e}$.

The idea of this algorithm is that in step 1, decoding of y using only C_0-decoding is attempted. If this fails then for some i' with $0 \leq i' \leq d-1$, we have $\mathrm{wt}(e_{i'}) > t_0$. But then by Lemma 4.3.9 (ii), for any $i > i'$, $\mathrm{wt}(e_i) \leq t_1$, and so we apply C_1-decoding starting from the highest degree term of y and working in the order of descending degrees.

Note that this algorithm, like Algorithm 4.3.5, is an instance of Procedure 4.3.1, and so Lemma 4.3.2 holds for the application of this algorithm. That is, at all steps, $\tilde{v}G + \tilde{e} = y$.

The next two results show that this algorithm correctly decodes y. Unless otherwise indicated, all references to steps in these two results refer to Algorithm 4.3.11. Most of the work is contained in the following technical lemma.

Lemma 4.3.12 *Suppose that for some fixed integer i' with $0 \leq i' < d$, $\mathrm{wt}(e_{i+1}) \leq t_1$ for all $i = d-1, d-2, \ldots i'$. Then for any $i = d-1, d-2, \ldots i'$, if step 3 is reached with that value of i, then we have:*

(i) At the beginning of step 3, $\tilde{e}_{i+1} = (v_i - \tilde{v}_i)G_1 + e_{i+1}$.

(ii) At the end of step 3, $\Delta v = (v_i - \tilde{v}_i)z^i$.

(iii) At the end of step 4, $\tilde{v}_i = v_i$ and $\tilde{e}_{i+1} = e_{i+1}$.

Moreover, in subsequent passes through the loop, these values of $\tilde{v}_i$ and $\tilde{e}_{i+1}$ remain unchanged.

Proof: First we prove (i), (ii), and (iii) by "backward" induction on i.

For our basis step we consider the pass through the loop with $i = d-1$. At the beginning of step 3 of this pass we have (as always) $\tilde{e} = y - \tilde{v}G$. Noting that in step 1 no G_0 terms get added to or subtracted from the z^d-term of $\tilde{e}$, we then have

$$\tilde{e}_d = y_d - \tilde{v}_{d-1}G_1 = (v_{d-1}G_1 + e_d) - \tilde{v}_{d-1}G_1 = (v_{d-1} - \tilde{v}_{d-1})G_1 + e_d.$$

So at the beginning of step 3 we have $i = d-1$ and

$$\tilde{e}_{i+1} = \tilde{e}_d = (v_{d-1} - \tilde{v}_{d-1})G_1 + e_d = (v_i - \tilde{v}_i)G_1 + e_{i+1}.$$

So (i) holds when $i = d-1$.

Then in step 3, because $\mathrm{wt}(e_d) = \mathrm{wt}(e_{i+1}) \leq t_1$, we apply C_1-decoding to $\tilde{e}_{i+1} = \tilde{e}_d$ and recover $v_{d-1} - \tilde{v}_{d-1} = v_i - \tilde{v}_i$. Hence at the end of step 3 we have $\Delta v = (v_{d-1} - \tilde{v}_{d-1})z^{d-1}$. So (ii) holds when $i = d-1$.

For (iii) we consider the situation just before entering step 4. At this moment we have

$$(\tilde{v} + \Delta v)_{d-1} = \tilde{v}_{d-1} + (v_{d-1} - \tilde{v}_{d-1}) = v_{d-1}.$$

Also at this moment we have

$$(\tilde{e}-(\Delta v)G)_d = \tilde{e}_d - (v_{d-1}-\tilde{v}_{d-1})G_1 = \tilde{e}_d + \tilde{v}_{d-1}G_1 - v_{d-1}G_1, \quad \text{but also}$$

$$e_d = y_d - w_d = (\tilde{v}G+\tilde{e})_d - v_{d-1}G_1 = (\tilde{e}_d + \tilde{v}_{d-1}G_1) - v_{d-1}G_1.$$

Therefore $(\tilde{e}-(\Delta v)G)_d = e_d$. So at the end of step 4 we will have $\tilde{v}_{d-1} = v_{d-1}$ and $\tilde{e}_d = e_d$. So (iii) holds when $i = d-1$.

Now we fix i_0 with $i' < i_0 \leq d-1$. We make the induction assumption that (i), (ii), and (iii) hold for $i = i_0$. In particular, $\tilde{v}_{i_0} = v_{i_0}$. We then must show that (i), (ii), and (iii) hold when $i = i_0 - 1$.

We consider the pass through the loop when $i = i_0 - 1$. At the beginning of step 3 of this pass we have (as always) $\tilde{e} = y - \tilde{v}G$. Therefore

$$\begin{aligned}
\tilde{e}_{i_0} &= y_{i_0} - (\tilde{v}G)_{i_0} = y_{i_0} - \tilde{v}_{i_0-1}G_1 - \tilde{v}_{i_0}G_0 \\
&= y_{i_0} - \tilde{v}_{i_0-1}G_1 - v_{i_0}G_0 = (vG+e)_{i_0} - \tilde{v}_{i_0-1}G_1 - v_{i_0}G_0 \\
&= (v_{i_0-1}G_1 + v_{i_0}G_0 + e_{i_0}) - \tilde{v}_{i_0-1}G_1 - v_{i_0}G_0 = (v_{i_0-1} - \tilde{v}_{i_0-1})G_1 + e_{i_0}.
\end{aligned}$$

Hence (i) holds when $i = i_0 - 1$.

Now in step 3, because $\mathrm{wt}(e_{i_0}) \leq t_1$ we apply C_1-decoding to $\tilde{e}_{i+1} = \tilde{e}_{i_0}$ and recover $v_{i_0-1} - \tilde{v}_{i_0-1} = v_i - \tilde{v}_i$. So at the end of step 3 we have $\Delta v = (v_{i_0-1} - \tilde{v}_{i_0-1})z^{i_0-1}$. So (ii) holds when $i = i_0 - 1$.

Hence just before we enter step 4 we have

$$(\tilde{v} + \Delta v)_{i_0-1} = \tilde{v}_{i_0-1} + v_{i_0-1} - \tilde{v}_{i_0-1} = v_{i_0-1}.$$

Also

$$(\tilde{e} - (\Delta v)G)_{i_0} = \tilde{e}_{i_0} + \tilde{v}_{i_0-1}G_1 - v_{i_0-1}G_1,$$

and also

$$\begin{aligned}
e_{i_0} &= y_{i_0} - w_{i_0} \\
&= (\tilde{v}G + \tilde{e})_{i_0} - v_{i_0-1}G_1 - v_{i_0}G_0 \\
&= (\tilde{v}_{i_0-1}G_1 + \tilde{v}_{i_0}G_0 + \tilde{e}_{i_0}) - v_{i_0-1}G_1 - v_{i_0}G_0 \\
&= (\tilde{v}_{i_0-1} - v_{i_0-1})G_1 + \tilde{e}_{i_0} \quad (\text{since } \tilde{v}_{i_0} = v_{i_0}).
\end{aligned}$$

So at the end of step 4 we will have $\tilde{v}_{i_0-1} = v_{i_0-1}$ and $\tilde{e}_{i_0} = e_{i_0}$. So (iii) holds when $i = i_0 - 1$.

So by induction (i), (ii), and (iii) hold beginning with $i = d-1$ and going backward as long as $\mathrm{wt}(e_{i+1}) \leq t_1$ and this holds through $i = i'$.

The final statement of the lemma follows because if $i_1 < i_2$, then when $i = i_1$, only $\tilde{v}_{i_1}$, $\tilde{e}_{i_1}$, and $\tilde{e}_{i_1+1}$ are changed. So $\tilde{v}_{i_2}$ and $\tilde{e}_{i_2+1}$ have indices too large to be changed when $i = i_1$. □

Proposition 4.3.13 *Algorithm 4.3.11 decodes y correctly (recovers v from y).*

Proof:

Case 1: $\mathrm{dist}(y_i, w_i) \leq t_0$ for all i, $0 \leq i < d$. Then by Proposition 4.3.8, $\tilde{v} = v$ at the end of step 1. Control is then transferred immediately to step 8, and the algorithm halts with y correctly decoded.

Case 2: $\mathrm{dist}(y_{i'}, w_{i'}) > t_0$ for some i' with $0 \leq i' < d$. Fix the least such i'. By Lemma 4.3.7, after step 1 is completed, even though $v \neq \tilde{v}$, we do have $\tilde{v}_i = v_i$ for all $i < i'$

Now by Lemma 4.3.9 (ii), for all $i > i'$ we must have $\mathrm{wt}(e_i) \leq t_1$. Equivalently for $i = i', i'+1, \ldots, d-1$ we have $\mathrm{wt}(e_{i+1}) \leq t_1$. Hence we may apply Lemma 4.3.12. After going through the loop of steps 3 through 7 for $i = d-1, d-2, \ldots, i'$ we have that for all such i, $\tilde{v}_i = v_i$. We observe that during these iterations of the loop, $\tilde{v}_i$ is not affected for any $i < i'$.

Hence when these iterations of the loop have been completed we have $\tilde{v}_i = v_i$ for $i = 0, 1, ..., d-1$. That is $\tilde{v} = v$. Then since $y = \tilde{v}G + \tilde{e} = vG + e$, we have $\tilde{e} = e$. Then $\mathrm{wt}(\tilde{e}) \leq t$. So at step 5 of the final time through the loop, control is transferred to step 8, and the algorithm halts with $\tilde{v} = v$. □

The following corollary gives an upper bound on how many times block-code decoding occurs.

Corollary 4.3.14 *In decoding y, C_0-decoding is applied at most d times, and C_1-decoding is applied at most d times. Hence block-code decoding occurs at most $2d$ times.*

Proof: The loop in step 1 of Algorithm 4.3.11 is performed at most for $i = 0, 1, \ldots, d-1$. Similarly the loop in steps 3 through 7 is performed at most for $i = d-1, d-2, \ldots, 0$. In a pass through either of these loops, one block-code decoding is attempted. The result follows immediately. $\square$

We conclude our discussion of the 1-D case for decoding with an example. We will use the binary field, $\mathbb{F} = \mathbb{F}_2$. For typographical reasons we will dispense with brackets ([ ]) and spaces between entries in writing elements of $\mathbb{F}^k$ and $\mathbb{F}^n$. So for instance, instead of

$$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} z + \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} z^2,$$

we write

$$110 + 111z + 010z^2.$$

Example 4.3.15 Let

$$G_0 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad G_1 = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

Note that the corresponding block codes C_0 and C_1 each consist of 4 elements:

$$C_0 = \{000000, 101100, 010111, 111011\}, \qquad C_1 = \{000000, 011011, 101110, 110101\}.$$

We see that $\delta_0 = 3$ and $\delta_1 = 4$. Also C_0 and C_1 may both be decoded by inspection.

We have

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} z.$$

Also $\delta = \delta_0 + \delta_1 = 7$, and $t_0 = \lfloor \frac{3-1}{2} \rfloor = 1,\ t_1 = \lfloor \frac{4-1}{2} \rfloor = 1,$ and $t = \lfloor \frac{7-1}{2} \rfloor = 3$.

Suppose we receive the word

$$y = 101100 + 100100z + 111001z^2 + 100110z^3 + 111011z^4 + 110101z^5.$$

We apply Algorithm 4.3.11 to y. Below we give a run history, keeping track of the quantities $i, \Delta v, \tilde{v}, \tilde{e}$, and $\mathrm{wt}(\tilde{e})$. In running the algorithm, the entries are obtained line by line, left to right. The horizontal line in the middle of the run shows where we have passed from the loop of step 1 to the loop of steps 3 through 7.

i	Δv	$\tilde{v}$	$\tilde{e}$	$\mathrm{wt}(\tilde{e})$
$-\infty$		00	$101100+100100z+111001z^2+100110z^3+111011z^4+110101z^5$	21
0	10	10	$000000+111111z+111001z^2+100110z^3+111011z^4+110101z^5$	22
1	$11z$	$10+11z$	$000000+000100z+001100z^2+100110z^3+111011z^4+110101z^5$	15
2	$10z^2$	$10+11z+10z^2$	$000000+000100z+100000z^2+111101z^3+111011z^4+110101z^5$	16
3	fail			
4	$11z^4$	$10+11z+10z^2+11z^4$	$000000+000100z+100000z^2+111101z^3+000000z^4+000000z^5$	7
3	$00z^3$	$10+11z+10z^2+11z^4$	$000000+000100z+100000z^2+111101z^3+000000z^4+000000z^5$	7
2	$11z^2$	$10+11z+01z^2+11z^4$	$000000+000100z+011011z^2+001000z^3+000000z^4+000000z^5$	6
1	$10z$	$10+01z+01z^2+11z^4$	$000000+101000z+000000z^2+001000z^3+000000z^4+000000z^5$	3

At this point $\mathrm{wt}(\tilde{e}) = 3 \leq t$. So the algorithm halts. The current value of $\tilde{v}$ is the original message word, v; and the current value of $\tilde{e}$ is the error, e. We may also recover the original codeword w, either by calculating vG, or by the short cut of subtracting the final value of $\tilde{e}$ (when $\tilde{e} = e$) from the initial value of $\tilde{e}$ (when $\tilde{e} = y$).

It is interesting to note that with hindsight we see that $i' = 1$ (where i' is as in the proof of Proposition 4.3.13), because the degree 1 term of e is the lowest degree term of e with weight greater than $t_0 = 1$. If we had known this in advance we could have decoded more efficiently, switching from the loop of step 1 to the loop of steps 3 through 7 immediately after $i = 0$. Then we would have successfully decoded y after just five decodings of block codes (as compared the eight decodings–including the one failure–that were performed in the actual run of the algorithm). Unfortunately there does not appear to be any way of determining i' in general before running the algorithm.

We now consider the general case of m-D decoding. So we are back in the situation where $G = G_0 + z_1G_1 + \cdots + z_mG_m$. We assume that the message word v is of total degree at most $d-1$, and so the received word, y, is of total degree at most d.

The key observation for decoding y is that the degree d part of y (consisting of the terms of y of total degree exactly d) may be regarded as a received word from an $(m-1)$-D unit memory code. Specifically if we denote by $v^{(i)}$, $y^{(i)}$, and $e^{(i)}$ the degree i parts of v, y, and e, we have

$$y^{(d)} = v^{(d-1)}(z_1G_1 + z_2G_2 + \cdots + z_mG_m) + e^{(d)}.$$

Note that in the sum involving the G_i, G_0 is not present, because G_0 does not contribute to the degree d part of y.

This observation allows us to inductively decode y based on being able to decode 1-D unit memory codes as discussed above.

Actually a double induction is involved, based on the degree of y as well as on the dimension of the code.

We adopt the following notation. $\epsilon_0 = (0,0,\ldots,0) \in \mathbb{N}^m$ is the zero element of $\mathbb{N}^m$, and $\epsilon_j = (0,\ldots,0,1,0,\ldots,0) \in \mathbb{N}^m$ (1 is in the j^{th} slot, $1 \le j \le m$). So for $j \ge 1$, ϵ_j is the j^{th} standard generating vector for $\mathbb{N}^m$.

The following algorithm decodes y in the case that v is of total degree at most 0 (i.e., $v = 0$ is possible), and y is of total degree at most $d = 1$. Note that in this case $v = v_{\epsilon_0}$. Then

$$y = \sum_{j=0}^{m} y_{\epsilon_j}\mathbf{z}^{\epsilon_j}, \quad \text{where } y_{\epsilon_j} = v_{\epsilon_0}G_j + e_{\epsilon_j}.$$

Algorithm 4.3.16

Input: y (here y is of total degree at most 1).

Step 1: $0 \longrightarrow j$.

Step 2: Attempt to C_j-decode $y_{\epsilon_j} = v_{\epsilon_0}G_j + e_{\epsilon_j}$. If unsuccessful, go to step 4. If successful, obtain $x \in \mathbb{F}^k$, an estimate for v_{ϵ_0}.

Step 3: If $\operatorname{dist}(xG, y) \le t$, go to step 6.

Step 4: $j + 1 \longrightarrow j$.

Step 5: Go to step 2.

Step 6: Stop. $v_{\epsilon_0} = x = v$. Successful decoding has occurred.

The following lemma shows that this algorithm successfully decodes y. We recall that the three assumptions preceding Remark 4.3.4 are in effect. In particular $\mathrm{wt}(e) \le t$.

Lemma 4.3.17 *If y is of total degree at most 1, then Algorithm 4.3.16 correctly decodes y.*

Proof: By Lemma 4.3.9 (ii), it cannot be the case for every $j, 0 \le j \le m$, that $\mathrm{wt}(e_{\epsilon_j}) > t_j$. That is, for some j' we have $\mathrm{wt}(e_{\epsilon_{j'}}) \le t_{j'}$. Then at step 2 of the algorithm with $j = j'$, correct decoding occurs. That is, the x obtained is in fact v_{ϵ_0}. Then in step 3, $\operatorname{dist}(xG, y) = \mathrm{wt}(e) \le t$. So control passes to step 6, and the algorithm halts with correct decoding of y. □

We are now ready to address the general situation where we are decoding a received word, y of total degree at most d ($d \geq 1$), from an m-dimensional unit memory code.

We can decode 1-D unit memory codes, and for any dimension we can decode received words of total degree 1. Therefore as induction hypotheses we may assume that we may decode multidimensional unit memory codes for dimension less than m and that we may decode any received word of degree less than d for a code of dimension m.

We apply Algorithm 4.3.5 to y, attempting to decode y using only C_0-decoding. If successful we are done. If unsuccessful, then we know by Proposition 4.3.8 that for some $\alpha \in \mathbb{N}^m$ with $|\alpha| < d$ that $\mathrm{wt}(e_\alpha) > t_0$.

We now consider $y^{(d)} = v^{(d-1)}(z_1G_1 + z_2G_2 + \cdots + z_mG_m) + e^{(d)}$. Consider the weight of $e^{(d)}$. Since $\mathrm{wt}(e) \leq t$ and since the total weight of the part of e of degree less than d is at least $t_0 + 1$ (since we are now in the case where Algorithm 4.3.5 failed to decode y), we have that $\mathrm{wt}(e^{(d)}) \leq t - t_0 - 1$. Hence by Lemma 4.3.9 (i) with $j = 0$, we have that

$$\mathrm{wt}(e^{(d)}) \leq \left\lfloor \frac{\delta_1 + \cdots + \delta_m - 1}{2} \right\rfloor .$$

Now $z_1G_1 + \cdots + z_mG_m$ generates a unit memory code of distance at least $\delta_1 + \cdots + \delta_m$. This code may be considered as being an $(m-1)$-dimensional unit memory code[1] because deleting the constant matrix, G_0, is clearly equivalent to deleting one of the matrices $G_j, j \geq 1$. Hence $y^{(d)}$, a received word from this code, has an error $e^{(d)}$, the weight of which is within the number of errors that can be corrected. Therefore $y^{(d)}$ is a received word of an $(m-1)$-dimensional code, and so by our induction hypothesis on dimension we may decode $y^{(d)}$.

Now in decoding $y^{(d)}$, we recover $v^{(d-1)}$ and $e^{(d)}$. Then we may form

$$y - v^{(d-1)}G - e^{(d)} = (vG + e) - v^{(d-1)}G - e^{(d)} = (v - v^{(d-1)})G + (e - e^{(d)}).$$

So $y - v^{(d-1)}G - e^{(d)}$ may be regarded as a received word of degree (at most) $d-1$. The error on this received word is $e - e^{(d)}$. And $\mathrm{wt}(e - e^{(d)}) \leq \mathrm{wt}(e) \leq t$. So by our induction hypothesis on degree, $y - v^{(d-1)}G - e^{(d)}$ may be decoded, yielding $v - v^{(d-1)}$.

Since we had previously recovered $v^{(d-1)}$, we may now form $(v - v^{(d-1)}) + v^{(d-1)} = v$. So y has been successfully decoded.

We conclude with an example.

Example 4.3.18 In this example, $\mathbb{F} = F_2$, and $R = \mathbb{F}[z_1, z_2, z_3]$.

$$G = \underbrace{\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}}_{G_0} + \underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}}_{G_1} z_1 + \underbrace{\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}}_{G_2} z_2 + \underbrace{\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}}_{G_3} z_3.$$

Then $\delta_0 = \delta_1 = \delta_3 = 2$; and $\delta_2 = 1$, and $t_0 = t_1 = t_2 = t_3 = 0$. So $\delta = 7$, and $t = 3$.

We will use notation similar to that in Example 4.3.15. We receive the word $y = 110 + 001z_2 + 001z_1z_2 + 011z_2^2 + 100z_2z_3 + 101z_3^2$. We have $d = 2$.

We begin by applying Algorithm 4.3.5 to y. Here is a run history. We omit the columns for α, $\tilde{v}$, and $\mathrm{wt}(\tilde{e})$. The current value of α can be deduced by looking at the monomial part of the current value of Δv; the current value of $\tilde{v}$ is the cumulative sum of the values of Δv up to the current value. Also we omit rows in which $\Delta v = 00\mathbf{z}^\alpha$, since in such a row there is no change in $\tilde{v}$ or $\tilde{e}$.

Δv	$\tilde{e}$
	$110 + 001z_2 + 001z_1z_2 + 011z_2^2 + 100z_2z_3 + 101z_3^2$
10	$101z_1 + 010z_2 + 011z_3 + 001z_1z_2 + 011z_2^2 + 100z_2z_3 + 101z_3^2$
$11z_1$	$010z_2 + 011z_3 + 011z_1^2 + 101z_1z_2 + 110z_1z_3 + 011z_2^2 + 100z_2z_3 + 101z_3^2$.

At this point Algorithm 4.3.5 fails because we cannot successfully C_0-decode 010.

[1] This code takes only homogeneous polynomials as message words (i.e., polynomials in which every term has the same total degree). One way to give this code the same format as the nonhomogeneous situation above is to dehomogenize $z_1G_1 + \cdots + z_mG_m$ at say the variable z_m (i.e., substitute the value 1 for z_m, obtaining the encoder $G_m + z_1G_1 + \cdots + z_{m-1}G_{m-1}$). Then also dehomogenize any message word at z_m. Then the situation is precisely the $(m-1)$-dimensional case.

So we now try to decode $y^{(2)} = 001z_1z_2 + 011z_2^2 + 100z_2z_3 + 101z_3^2$ as a received word from the code generated by $z_1G_1 + z_2G_2 + z_3G_3$. We remark that the distance lower bound for this code is $\delta_1 + \delta_2 + \delta_3 = 2 + 1 + 2 = 5$, so we must find a codeword with distance at most 2 from $y^{(2)}$.

To decode $y^{(2)}$, we must apply a straightforward variant of Algorithm 4.3.5 using C_1-decoding. But C_1-decoding fails immediately because we cannot C_1-decode the coefficient 001 of the leading z_1-term.

So now we must try to decode the degree 2 part of y that does not involve the indeterminate z_1. We call this part $y^{(2')}$. So $y^{(2')} = 011z_2^2 + 100z_2z_3 + 101z_3^2$. Here the decoding is with respect to the code generated by $z_2G_2 + z_3G_3$, for which the distance lower bound is $\delta_2 + \delta_3 = 2 + 1 = 3$. So this code corrects 1 error. We apply Algorithm 4.3.5 with C_2 decoding. Here is the run history.

$\Delta v^{(1')}$	$\widetilde{e^{(2')}}$
	$011z_2^2 + 100z_2z_3 + 101z_3^2$
$10z_2$	$111z_2z_3 + 101z_3^2$
$01z_3$	$000.$

Let $v^{(1')} = 10z_2 + 01z_3$. Thus $v^{(1')}(z_2G_2 + z_3G_3) = y^{(2')}$. The error term here is zero since at the end of the above run, $\widetilde{e^{(2')}} = 000$.

So we form $y^{(2)} - v^{(1')}(z_1G_1 + z_2G_2 + z_3G_3)$. ($G_0$ does not appear because we are still in the process of decoding $y^{(2)}$ with respect to $z_1G_1 + z_2G_2 + z_3G_3$). We have

$$y^{(2)} - v^{(1')}(z_1G_1 + z_2G_2 + z_3G_3) = 100z_1z_2 + 110z_1z_3.$$

Call this word $y^{(2'')}$.

We apply Algorithm 4.3.5 to $y^{(2'')}$ with C_1-decoding (this is correct because we have reduced the degree of z_1 from 2 in $y^{(2)}$ to 1 in $y^{(2'')}$. Here is the run history.

$\Delta v^{(1'')}$	$\widetilde{e^{(2'')}}$
	$100z_1z_2 + 110z_1z_3$
$11z_1$	$011z_1^2.$

This completes the decoding of $y^{(2)}$. We obtain $v^{(1)} = 11z_1 + 10z_2 + 01z_3$ and $e^{(2)} = 011z_1^2$.

So we form $y - v^{(1)}G - e^{(2)} = 110 + 101z_1 + 111z_2 + 011z_3$. We try to decode this word with respect to G. The decoding is successful–we obtain 10 as the corresponding message word with an error of $100z_2$.

Putting all this information together, we have $v = 10 + 11z_1 + 10z_2 + 01z_3$ and $e = 100z_2 + 011z_1^2$. Then $y = vG + e$ with $\mathrm{wt}(e) = 3 \leq t$. Decoding of y has been successfully completed.

CHAPTER 5

CATASTROPHICITY

5.1 Catastrophic Codes

The notion of catastrophicity for 1-D codes was first introduced by J. L. Massey at a conference. Apparently the first development of catastrophicity in the literature is in the paper by Massey and Sain [30]. A catastrophic code is one in which an infinite weight input (message word) produces a finite weight output (codeword). In this section we carefully define catastrophicity in the m-D case and derive a necessary and sufficient condition for a code to be catastrophic. Fornasini and Valcher consider catastrophicity for 2-D codes. The condition they give for catastrophicity is not the same as the one found here because they are working over the Laurent polynomial ring (see Remark 5.1.8 at the end of this section).

Let $\hat{R} = \mathbb{F}[[z_1, \ldots, z_m]]$ be the power series ring in m indeterminates over $\mathbb{F}$. For $p \in \hat{R}$ we say p has *infinite weight* if $p \notin R$ (so p has infinitely many nonzero terms). We write $\mathrm{wt}(p) = \infty$. Also for $v \in \hat{R}^k \setminus R^k$ or $w \in \hat{R}^n \setminus R^n$ we write $\mathrm{wt}(v) = \infty$ or $\mathrm{wt}(w) = \infty$.

For $f \in R$ and $p \in \hat{R}, pf$ is defined by ordinary multiplication of a power series times a polynomial (we will not use the left-shift action of R on $\hat{R}$ from Section 2.6).

Let $G \in R^{k\times n}$. As usual $\mathcal{C}(G) = \mathrm{rowspace}_R(G) \subseteq R^n$ is the code generated by G. We further define $\hat{\mathcal{C}}(G) = \mathrm{rowspace}_{\hat{R}}(G) \subseteq \hat{R}^n$. (So $\hat{\mathcal{C}}(G)$ may be thought of as a code in which some of the message words and codewords have infinite weight.)

Definition 5.1.1 The code $\mathcal{C}(G)$ is *catastrophic* if there exists $v \in \hat{R}^k$ with $\mathrm{wt}(v) = \infty$ and $\mathrm{wt}(vG) < \infty$. That is some infinite weight message words produce finite weight codewords.

Equivalently $\mathcal{C}(G)$ is catastrophic if $(\hat{\mathcal{C}}(G) \setminus \mathcal{C}(G)) \cap R^n \neq \phi$.

Another way to think of this is that $\mathcal{C}(G)$ is noncatastrophic if the set of finite weight codewords of $\hat{\mathcal{C}}(G)$ is precisely $\mathcal{C}(G)$.

Note that although the condition of catastrophicity is stated in terms of $\hat{\mathcal{C}}(G)$, the code $\mathcal{C}(G)$ is the catastrophic code.

Catastrophic codes are undesirable in that a finite number number of transmission errors may cause an infinite number of decoding errors. For instance suppose that $v \in \hat{R}^k$ with $\mathrm{wt}(v) = \infty$ and suppose that $\mathrm{wt}(vG) = j < \infty$. Then j errors in transmission could change the codeword $w = vG$ to the received word $y = 0$. Then decoding y would yield the estimated message word $\tilde{v} = 0$. But $\mathrm{dist}(v, \tilde{v}) = \mathrm{dist}(v, 0) = \mathrm{wt}(v) = \infty$. Thus infinitely many decoding errors have occurred.

Here is an example of a catastrophic code.

Example 5.1.2 Consider the 1-D rate $\frac{1}{2}$ code over the binary field $\mathbb{F}_2$ with generator matrix $G = [1 + z + z^2 \quad 1 + z + z^2]$. Let

$$v = 1 + z + z^3 + z^4 + z^6 + z^7 + \cdots = \sum_{i=0}^{\infty} (z^{3i} + z^{3i+1}).$$

Then $vG = [1 \ \ 1]$. So $\mathrm{wt}(v) = \infty$ but $\mathrm{wt}(vG) = 2$. So two errors in transmission could cause 0 to be recovered from decoding instead of v.

The following result for the 1-D case is well known. (See Massey and Sain [30] or Forney [12]. Also this result is a special case of Proposition 5.1.7 below.)

Proposition 5.1.3 *Let $R = \mathbb{F}[z]$. Let $G \in R^{k\times n}$ be of rank k. So G generates a 1-D convolutional code, $\mathcal{C}(G)$. $\mathcal{C}(G)$ is noncatastrophic if and only if the greatest common divisor of the full-size minors of G is of the form z^l for some integer $l \geq 0$.*

We next give an example to show that in the general m-D case a monomial greatest common divisor of the full-size minors of G is not a necessary condition for $\mathcal{C}(G)$ to be noncatastrophic.

Example 5.1.4 Consider the rate $\frac{1}{1}$ 2-D code over $R = \mathbb{F}_2[z_1, z_2]$ generated by $G = [z_1 + z_2]$. The only full-size minor of G is $z_1 + z_2$, not a monomial. But we claim that $\mathcal{C}(G)$ is noncatastrophic. To see this suppose that $p \in \hat{R}$ with $\mathrm{wt}(p) = \infty$. We must show that $\mathrm{wt}(pG) = \infty$.

Decompose p by total degree: $p = p^{(0)} + p^{(1)} + p^{(2)} + \cdots$ where $p^{(i)}$ is a homogeneous polynomial of degree i (i.e., all the terms have total degree i). Then $pG = p^{(0)}G + p^{(1)}G + \cdots$ gives a decomposition of pG by total degree ($p^{(i)}G$ is the degree $i+1$ part of pG). Now for any i with $p^{(i)} \neq 0$ we have $p^{(i)}G \neq 0$. Thus for such i, $\mathrm{wt}(p^{(i)}G) > 0$. Since $\mathrm{wt}(p) = \infty$, infinitely many of the $p^{(i)}$ have positive weight. So infinitely many of the $p^{(i)}G$ have positive weight. Then

$$\mathrm{wt}(pG) = \sum_{i=0}^{\infty} \mathrm{wt}(p^{(i)}G) = \infty.$$

We conclude that $\mathcal{C}(G)$ is noncatastrophic as claimed.

The following corollary is a trivial rephrasing of Proposition 5.1.3. The reason that we state it is that it leads to the correct generalization of the proposition to the m-D case.

Corollary 5.1.5 *Let R and G be as in Proposition 5.1.3 (1-D case). $\mathcal{C}(G)$ is catastrophic if and only if there exists a nonconstant polynomial $\varphi \in R$ with nonzero constant term such that φ is a common divisor of all the full-size minors of G.*

Next we consider conditions for catastrophicity in the m-D case. So now $R = \mathbb{F}[z_1, ..., z_m]$, and $\hat{R} = \mathbb{F}[[z_1, ..., z_m]]$.

Lemma 5.1.6 *Let $v = [v_1, ..., v_k] \in R^k, d \in R$ with d not a monomial , and $p = [p_1, ..., p_k] \in \hat{R}^k$. Suppose that $v_1, ..., v_k, d$ have no nonunit common factors in R. If $p\,d = v$, then d has a nonzero constant term.*

The following proof of the lemma was derived jointly with M. Anton.

Proof: Fix some monomial order on monomials in the indeterminates $z_1, ..., z_m$. Let $\mathrm{TT}(d)$ be the trailing term of d with respect to this monomial order.

We claim that z_1 does not divide $\mathrm{TT}(d)$. (Note that once this has been shown, the same argument will show that z_j does not divide $\mathrm{TT}(d)$ for $1 \leq j \leq m$. Then $\mathrm{TT}(d)$ is a constant, and by definition nonzero. This will establish the lemma). Suppose to the contrary that z_1 divides $\mathrm{TT}(d)$. Now consider the ring $\tilde{R} = \mathbb{F}(z_2, ..., z_m)[z_1]$ of polynomials in one indeterminate, z_1, over the field of rational functions, $\mathbb{F}(z_2, ..., z_m)$.

Considered as elements of $\tilde{R}$, $v_1, ..., v_k, d$ have no nonunit common factors in $\tilde{R}$ (this standard result can be found for instance, for $k = 1$, in Cox, Little, and O'Shea [3, p. 149]). Hence there exists $u_1, ..., u_k, f \in R$, and $g \in \mathbb{F}[z_2, ..., z_m]$ such that

$$v_1 \frac{u_1}{g} + \cdots + v_k \frac{u_k}{g} + d\frac{f}{g} = 1.$$

Putting $u = [u_1, ..., u_k] \in R^k$, we have $v\,u^t + d\,f = g$. Hence $d(p\,u^t + f) = g$. Now z_1 appears in $\mathrm{TT}(d)$. Hence z_1 appears in the trailing term of $d(p\,u^t + f)$. But z_1 does not appear in g. This is a contradiction. Hence the claim is established, and as remarked above, this proves the lemma. □

The following proposition generalizes Corollary 5.1.5 to the m-D case.

Proposition 5.1.7 *Let $G \in R^{k\times n}$ be of rank k. The code $\mathcal{C}(G)$ is catastrophic if and only if there exists a nonconstant polynomial $d \in R$ with nonzero constant term such that d is a common divisor of all the full-size minors of G.*

Proof: First we assume the existence of such a polynomial d. We may assume that the constant term of d is 1 (if necessary replace d by $d_0^{-1}d$ where $d_0 \in \mathbb{F} \setminus \{0\}$ is the constant term of d. This will not affect the full-size minors of G being divisible by d). Observe that d is a unit in $\hat{R}$, for since the constant term of d is 1, $\frac{1}{d}$ may be expanded into a power series using the geometric series.

Now by Corollary 3.3.15, there exists $v \in R^k$ such that d divides every entry of vG, but d does not divide every entry of v. Hence $\frac{v}{d}$ is an infinite weight element of $\hat{R}^k$.

We have that $\frac{v}{d}G \in R^k$ since d divides every entry of vG. We conclude that $\mathcal{C}(G)$ is catastrophic because the infinite weight message word $\frac{v}{d}$, is encoded as the finite weight codeword $\frac{v}{d}G$.

To prove the converse we assume that $\mathcal{C}(G)$ is catastrophic. So there exists $p \in \hat{R}^k$ such that $\mathrm{wt}(p) = \infty$ and $\mathrm{wt}(pG) < \infty$ (i.e., $pG \in R^n$).

Now since G is of full row rank, there exists $G' \in R^{n\times k}$ and $d \in R$ $(d \neq 0)$ such that $G\,G' = d\,I_k$. Let $w = pG \in R^n$. Then we have $p\,d = pG\,G' = wG' \in R^k$. Let $v = wG' \in R^k$. So $p\,d = v$. Writing $v = [v_1\,v_2\,\ldots\,v_k]$, we may assume that $v_1, ..., v_k, d$ have no nonunit common factors in R (if they do have a common factor, it may be cancelled in the equation $p\,d = v$).

So we have $vG = d\,pG = d\,w$. Then by Corollary 3.3.11, d divides every full-size minor of G.

Now since $p\,d = v$ with p an infinite weight element of $\hat{R}^k$ and v a (finite weight) element of R^k, we have that $d \in R$ is not a monomial. Then since $v_1, ..., v_k, d$ have no nonunit common factors, by Lemma 5.1.6, d has a nonzero constant term.

So d is the polynomial required in the statement of the proposition. □

Remark 5.1.8 It is worth observing that the above result depends crucially on the ring under consideration. In particular, over the Laurent polynomial ring, $\mathbb{F}[z_1, z_1^{-1}, ..., z_m, z_m^{-1}]$, the condition of noncatastrophicity is equivalent to the full size minors of G having only monomial common divisors, which would give a different generalization of the 1-D notion of catastrophicity. As mentioned in Section 1.1, Fornasini and Valcher [8] look at 2-D convolutional codes using the Laurent polynomial ring, and in this paper they consider catastrophicity of codes over this ring.

CHAPTER 6

CONCLUSION

6.1 Summary

In this dissertation we have introduced multidimensional convolutional codes from a module-theoretic point of view. In Chapters 2, 3, and 5 we considered some of the basic issues of m-D codes such as free and nonfree codes, generator matrices and encoders, weight and distance, duality with m-D autoregressive behavioral systems, catastrophicity, and algebraic structure and representation. With regard to this last, we observed that any m-D convolutional code has an image representation, and we gave conditions for a code to have a kernel representation. Largely because of the more complicated structure of a polynomial ring in several indeterminates compared to a polynomial ring in one indeterminate, m-D convolutional codes are a nontrivial generalization of 1-D convolutional codes. We noted many of the differences among the 1-D case, the 2-D case, and higher dimensional cases. For instance a 1-D code is always free; a 2-D code always has an orthogonal code that is free; and a 2-D code is always contained in a 2-D free code of the same rank.

In Chapter 4, we gave a code construction for a class of codes. Using a new application of monomial orders, we derived a lower distance bound for codes in this class. We also developed a decoding algorithm for codes in this class.

6.2 Future Directions

There are many unanswered questions and many possible directions for research in the new area of multidimensional convolutional codes. We plan to address some of these issues in future work. Also we hope others will take an interest in this field. In this section we give several possibilities for further study that arose in working on this dissertation.

(i). Strengthen the connection between m-D codes and m-D systems. The duality developed by Oberst and sketched in Section 2.6 gives a method for translating and applying results of systems theory to coding theory and vice versa. This program is being actively carried out in the 1-D case (see for instance [55, 44, 43]).

(ii). Further work on distance and distance bounds for m-D convolutional codes should be undertaken. A great deal is known about distance bounds for 1-D convolutional codes (see for instance [21, 2]). Also in the 1-D case there are finite time algorithms available for computing the distance of a convolutional code (these algorithms exhibit exponential growth and so are of limited practical value, but they are useful for codes of low complexity, and they are of theoretical interest). Currently there is no finite time algorithm available for calculating the distance of an m-D convolutional code for $m \geq 2$. We found two obstacles in generalizing 1-D techniques. In the 1-D case there is one monomial of every degree. In the m-D case, the number of monomials increases with degree (in fact it is not difficult to show that there are $\binom{m+d-1}{m-1}$ monomials of degree d in the case of m indeterminates). Because of this growth of monomials, graph theoretic methods for calculating distance fail. Another problem in computing distance using a method based on having a parity check matrix available arose because in the 1-D case there are minimum weight codewords whose constant term is nonzero, but in the m-D case this is not so.

(iii). We do not have specific applications of m-D codes at this time. It seems plausible that m-D codes will be found useful in transmitting m-D data. For instance 2-D codes for pictures and video images, 3-D codes for tomographic and MRI applications. It is our understanding[1] that the way convolutional codes are currently employed, there is, before the coding step, a compression step. At this point redundancy is squeezed out of the message before adding redundancy at the encoding step. In actual applications of m-D convolutional codes, it would be very good to use a compression method that in some sense preserves the multidimensionality of the data. Developing such compression techniques would be a worthwhile endeavor.

(iv). Implementation of encoding and decoding for m-D codes is needed. 1-D codes are implemented using shift registers. Input information (terms of the message word) go into shift registers. After passing through in a certain fixed number of steps, the input information may be discarded. The amount of information being held in the shift registers is time-independent. In the m-D case because of the increase in number of monomials with degree (see (ii) above), the amount of information that would have to be held in a shift register implementation would increase with time. This is a serious problem where for long (high degree) messages, the amount of information that is needed at a given time gets very large. In the 2-D case, this issue is related to the question of state for 2-D systems, which is considered by Roesser [41], by Fornasini and Marchesini [7], by Rocha [38], and by Rocha and Willems [39, 40].

(v). Find relevant good forms and invariants for m-D convolutional codes. In the 1-D case, so-called column-proper form for a generator matrix of a code [55] gives certain invariants for the code (i.e., column degrees–also called Kronecker indices in the systems theory literature–and McMillan degree). What are the good forms for a generator matrix of an m-D convolutional code? What are the invariants? We remark that for free convolutional codes, the full-size minors of an encoder (i.e., a full row rank generator matrix) form a set of projective invariants. We do not know if they are a complete set of invariants (i.e., whether the code may be recovered from the invariants).

(vi). Strengthen the connection between m-D convolutional codes and algebraic geometry. Connections between 1-D systems theory and algebraic geometry have been explored (see for instance [37, 29]). Because of the duality between codes and systems, this gives a connection between 1-D convolutional codes and quotient sheaves over the projective line. Because of the development of the theory of m-D convolutional codes using polynomial matrices and modules, there are many connections between m-D convolutional codes and algebraic geometry that should be explored.

(vii). There may be a connection between various matrix primeness notions and vector bundles on projective space. For instance, the cotangent bundle on $\mathbb{P}^2$ may be represented by the matrix

$$\begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \\ -z_3 & z_2 & 0 \end{bmatrix}.$$

The alert reader may recognize this matrix and its upper 2×3 submatrix as being used as an important example in this dissertation (see Proposition 3.1.7, Example 3.3.13, and Example 3.4.8). The author and C. Peterson are currently exploring the connection between matrices that are factor prime but not minor prime and vector bundles. It is their hope to find new vector bundles through such matrix considerations.

The above list should by no means be taken to be exhaustive of the subject. As this area is further studied, many more interesting results and questions will come to light.

[1] This is based on conversations with O. Takeshita

BIBLIOGRAPHY

[1] W.W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases*, volume 3 of *Graduate Studies in Mathematics*. American Mathematical Society, Rhode Island, 1994.

[2] D. Costello. Free distance bounds for convolutional codes. *IEEE Trans. Inform. Theory*, IT-20(3):356–365, 1974.

[3] D. Cox, J. Little, and D. O. O'Shea. *Ideals, Varieties and Algorithms*. Undergraduate Text in Mathematics. Springer Verlag, New York, 1992.

[4] D. Eisenbud. *Commutative algebra with a view toward algebraic geometry*. Graduate texts in mathematics. Springer-Verlag, New York, Berlin, 1994.

[5] D.R. Estes and J.R. Matijevic. Matrix factorizations, exterior powers, and complete intersections. *J. Algebra*, 58(1):117–135, 1979.

[6] D.R. Estes and J.R. Matijevic. Unique factorization of matrices and Towber rings. *J. Algebra*, 59(2):387–394, 1979.

[7] E. Fornasini and G. Marchesini. State–space realization theory of two–dimensional filters. *IEEE Trans. Automat. Contr.*, AC–21(4):484–492, 1976.

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

[9] E. Fornasini and M.E. Valcher. Observability and extendability of finite support nD behaviors. In *Proc. of the 34th IEEE Conference on Decision and Control*, pages 3277–3282, New Orleans, Louisiana, 1995.

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

[11] 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.

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

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

[14] G. D. Forney. Convolutional codes II: Maximum likelihood decoding. *Inform. Control*, 25:222–266, 1974.

[15] P. A. Fuhrmann. *A Polynomial Approach to Linear Algebra*. Universitext. Springer-Verlag, New York, 1996.

[16] W. Greub. *Linear algebra, 4th ed.* Graduate Texts in Math. No 23. Springer, New York-Berlin-Heidelberg, 1974.

[17] R. Hill. *A First Course in Coding Theory*. Oxford Applied Math. & Comp. Science Series, 1988.

[18] T.W. Hungerford. *Algebra*. Springer, New York, 1980.

[19] I.M. Isaacs. *Algebra. A graduate course*. Brooks/Cole, Pacific Grove, California, 1994.

[20] N. Jacobson. *Basic Algebra I*. W.H. Freeman and Co., San Francisco, 1974.

[21] R. Johannesson and K. Zigangirov. Distances and distance bounds for convolutional codes – an overview. In *Topics in Coding Theory. In honour of L. H. Zetterberg.*, Lecture Notes in Control and Information Sciences # 128, pages 109–136. Springer Verlag, 1989.

[22] G. Köthe. *Topological Vector Spaces I*. Springer Verlag, 1969.

[23] T.Y. Lam. *Serre's Conjecture*. Springer-Verlag, Berlin Heidelberg, 1978.

[24] S. Lang. *Algebra* (3rd ed.). Addison-Wesley, Reading, Mass., 1993.

[25] L. N. Lee. Short unit-memory byte-oriented binary convolutional codes having maximal free distance. *IEEE Trans. Inform. Theory*, IT-22(3):349–352, 1976.

[26] S. Lin and D. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[27] F.S. Macaulay. *The Algebraic Theory of Modular Systems*. Cambridge University Press, Cambridge, 1916.

[28] F. J. MacWilliams and N. J.A. Sloane. *The Theory of Error-Correcting Codes*. North Holland, Amsterdam, 1977.

[29] C. F. Martin and R. Hermann. Applications of algebraic geometry to system theory: The McMillan degree and Kronecker indices as topological and holomorphic invariants. *SIAM J. Control Optim.*, 16:743–755, 1978.

[30] J. L. Massey and M. K. Sain. Inverses of linear sequential circuits. *IEEE Trans. on Computers*, C-17(4):330–337, 1968.

[31] 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.

[32] D. G. Northcott. *Finite free resolutions*. Cambridge Tracts in Mathematics 71. Cambridge University Press, Cambridge, 1973.

[33] U. Oberst. Multidimensional constant linear systems. *Acta Appl. Math*, 20:1–175, 1990.

[34] C. Peterson, J. Rosenthal, and P. A. Weiner. Connections between multidimensional systems theory and algebraic geometry. In *Proc. of the 34-th Allerton Conference on Communication, Control, and Computing*, pages 583–592, 1996.

[35] Ph. Piret. *Convolutional Codes, an Algebraic Approach*. MIT Press, Cambridge, MA, 1988.

[36] Ph. Piret. A convolutional equivalent to Reed-Solomon codes. *Philips J. Res.*, 43(3-4):441–458, 1988.

[37] M. S. Ravi, J. Rosenthal, and J. M. Schumacher. Homogeneous behaviors. *Math. Contr., Sign., and Syst.*, 10:61–75, 1997.

[38] M.P.M Rocha. *Structure and Representation of 2D Systems*. PhD thesis, University of Groningen, 1990.

[39] P. Rocha and J. C. Willems. State for 2-D systems. *Linear Algebra Appl.*, 122/123/124:1003–1038, 1989.

[40] P. Rocha and J. C. Willems. Canonical computational forms for ar 2-D systems. *Multidimensional Systems and Signal Processing*, 2:251–278, 1990.

[41] R.P. Roesser. A discrete state–space model for linear image processing. *IEEE Trans. Automat. Contr.*, AC–20:1–10, 1975.

[42] S. Roman. *Coding and Information Theory.* Graduate Texts in Mathematics. Springer Verlag, New York - Berlin, 1992.

[43] J. Rosenthal, J. M. Schumacher, and E.V. York. On behaviors and convolutional codes. *IEEE Trans. Inform. Theory*, 42(6):1881–1891, 1996.

[44] J. Rosenthal and R. Smarandache. Construction of convolutional codes using methods from linear systems theory. In *Proc. of the 35-th Annual Allerton Conference on Communication, Control, and Computing*, pages 953–960, 1997.

[45] C.E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:379–423 and 623–656, 1948.

[46] C. Thommesen and J. Justesen. Bounds on distances and error exponents of unit memory codes. *IEEE Trans. Inform. Theory*, IT-29(5):637–649, 1983.

[47] M.E. Valcher and E. Fornasini. On 2D finite support convolutional codes: an algebraic approach. *Multidim. Sys. and Sign. Proc.*, 5:231–243, 1994.

[48] J.L. Walker. *Algebraic geometric codes over rings.* PhD thesis, University of Illinois at Urbana-Champaign, 1996.

[49] J. C. Willems. Deducing the input/output and the input/state/output structure from the external behavior. In *Proc. 25th IEEE Conf. Dec. Contr.*, pages 1936–1937, New York, 1986. IEEE Press.

[50] J. C. Willems. From time series to linear system. Part I: Finite dimensional linear time invariant systems. *Automatica*, 22:561–580, 1986.

[51] J. C. Willems. Models for dynamics. In U. Kirchgraber and H. O. Walther, editors, *Dynamics Reported*, volume 2, pages 171–269. John Wiley & Sons Ltd, 1989.

[52] J. C. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Trans. Automat. Control*, AC-36(3):259–294, 1991.

[53] J. Wood, E. Rogers, and D.H. Owens. A behavioral approach to the pole structure of 1D and nD linear systems. *Preprint*, 1997.

[54] J. Wood, E. Rogers, and D.H. Owens. A formal theory of matrix primeness. *Preprint*, 1997.

[55] E.V. York. *Algebraic Description and Construction of Error Correcting Codes, a Systems Theory Point of View.* PhD thesis, University of Notre Dame, 1997. Available at http://www.nd.edu/~rosen/preprints.html.

[56] D. C. Youla and G. Gnavi. Notes on n-dimensional system theory. *IEEE Trans. Circuits and Systems*, 26:105–111, 1979.

[57] 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.

[58] J. Zeng. A bijective proof of Muir's identity and the Cauchy-Binet formula. *Linear Algebra Appl.*, 184:79–82, 1993.

[59] E. Zerz. Primeness of multivariate polynomial matrices. *Systems & Control Letters*, 29:139–145, 1996.