CODING THEORY AND ALGEBRAIC CURVES

by

BRANDON GARY EDWARDS

B.A. (Mathematics, Physics) Portland State University,(June 1993)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

Department of Mathematics

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

April 1997

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Mathematics
The University of British Columbia
Vancouver, Canada

Date April 29, 1997

Abstract

In 1981 V.D. Goppa [9] used the evaluation of rational functions on algebraic curves to define a new and exciting class of error correcting codes now known as geometric Goppa codes. As with any other error correcting code however, this construction procedure alone was only the beginning of what was need in order that these codes could be used in practice. Procedures needed to be found to aid in both the explicit construction of certain geometric Goppa codes and the creation of efficient decoding algorithms. The problems surrounding these concerns have since motivated many coding theorists to become actively involved in the study of algebraic curves and have equally sparked the interests of algebraic geometers.

In this paper I introduce the topic of coding theory and highlight the successes and failures that have occurred in the attempt to bring geometric Goppa codes to their full stage of implementation. In the first two chapters, I give a basic introduction to the theory of error correcting codes, assuming no previous knowledge of the subject. Chapter 3 is concerned with the construction of rational geometric Goppa codes for which no knowledge of algebraic curves is necessary. Various positive and negative aspects of these codes are discussed which motivates the introduction of the general class of geometric Goppa codes in the following two chapters. For the material in Chapters 4 and 5, I will assume that the reader is familiar with divisors and linear systems on non-singular projective curves. After I define the class of geometric Goppa codes, I go on to discuss the advances that have been made in the search for explicit constructions of some of the best codes in this class. Finally, I present some basic decoding algorithms for both rational and general geometric Goppa codes. These algorithms demonstrate how easy it is to develop efficient decoders in both of these cases.

Table of Contents

Acknowledgment

The basic ideas concerning geometric Goppa codes that I have expressed in this paper, (including the concepts and algorithm constructions of Chapter 5), were developed through reading a text on algebraic function fields and coding theory written by Henning Stichtenoth [4]. For the material in Section 1.4, I am indebted to Joel Friedman for many clarifying discussions. Finally, for editing and guidance I would like to thank Bill Casselman, Kai Behrend, and Anne O'Halloran.

Chapter 1
Error Correcting Codes

1.1 The Problem

Consider a battery powered space satellite orbiting a nearby planet and collecting data. Every hour the satellite observes whether or not a certain phenomenon has occurred and records the answer of 1 or 0, ('yes' or 'no'), in its memory. This satellite will never return to earth, so at some point the information must be transmitted back to its home country. To do so, it will pass the information to a battery-powered relay satellite halfway to earth. The relay satellite will then transmit the received message home. The problem lies in that the satellites are using weak transmission signals. There is a certain probability, both at the relay satellite and at home, that a transmitted 1 will be received as a 0 and an equal probability that a 0 will be received as a 1. We wish to find an agreed upon 'coding' of the information that will minimize the probability of miscommunication while not wasting too much battery power in the process of sending and 'decoding' the messages.

As a first attempt, we could use what is called a 'repetition code'. In this case every hour, the first satellite would record whether or not the phenomenon occurred and transmit the data, 'encoded' as say 101 repetitions of the appropriate signal. Therefore it will transmit either 1...1 or 0...0. The relay satellite would then simply count the number of 1's and 0's in the received 101-tuple and 'decode' the message by taking the more frequently occurring symbol as the one intended. The decoded message, (either 1...1 or 0...0), would then be sent home, which would again be decoded upon receipt. Increasing the number of repetitions to 201, 401, and more; we could lower the probability of having an incorrect message after decoding to

almost zero. However, the battery power required to transmit these lengthy messages as well as the increase power consumption due to the counting procedures needed to decode them would drain the batteries too quickly. The ratio of the number of signals required to send the original messages to number of signals used is called the *information rate* of a code. Therefore codes with lower information rates use up more time and energy and the problem with the repetition codes can be stated in the following way. Any repetition code whose corresponding probability of incorrect message reception is small will necessarily have a small information rate, therefore consuming too much time and energy.

We are certainly forced to build the satellites so that their battery power can support the transmission of raw data for the duration of the mission, however it is clearly advantageous to keep the extra signals used for error control to a minimum.

The development of the theory of error correcting codes depends mainly on constructing codes, $\mathbf{C}$, which maximize the information rate and minimize both the probability of an incorrect message after decoding, ($P_{\mathbf{C}}$), and labour in decoding.

1.2 Basic Definitions

We will base our analysis on a few practical assumptions. First of all, we will assume that information is being transmitted from one location to another as words of length n in an alphabet of q signals. To allow for the introduction of a field structure, (the specifics of which will be seen later), we assume that $q = p^m$ for some integer m and prime p. Finally, due to week signal strength or some other physical problem, the received words will periodically contain some incorrect signals. We will assume that the probability of an error occurring while transmitting a signal is the same no matter what signal is being used and that the incorrect signal that is received can be any one of the $q - 1$ remaining signals with equal probability.

We will start with a collection of vectors selected from the space $\mathbb{F}_q^n$ where $\mathbb{F}_q$ denotes the finite field of order q. Every vector in this collection will represent a different message. In the case of the repetition code of length 101, this collection was $\{(0\ldots0), (1\ldots1)\} \subseteq \mathbb{F}_2^{101}$.

In addition, we require tools to aid in one of the most important aspects of our analysis: the determination of $P_{\mathbf{C}}$. The value of $P_{\mathbf{C}}$ depends upon how many entries need to be changed in any given codeword to convert it into another codeword. The specific field entries that are involved in these changes are irrelevant due to our assumption on signal error above. In order to quantify these differences among codewords, we introduce a distance μ to $\mathbb{F}_q^n$ given by

$$\mu(\mathbf{x}, \mathbf{y}) = \{ \text{ number of positions in which } \mathbf{x} \text{ and } \mathbf{y} \text{ differ } \}.$$

It can be easily verified that $(\mathbb{F}_q^n, \mu)$ is in fact a metric space.

Before now, we have used the word 'code' in a vary loose sense. We can now introduce a precise definition.

Definition 1 *A code, $\boldsymbol{C}$, is a subset of the metric space $(\mathbb{F}_q^n, \mu)$ for some n.*

To decode a given received vector $\mathbf{x}$, we must then find a codeword, $\mathbf{c}$, that is closest to $\mathbf{x}$ with respect to the distance μ. In the case that the choice of $\mathbf{c}$ is not unique, we are forced to randomly select one out of the collection of 'closest codewords', all of which are equally likely to have been the intended message.

One characteristic of $\mathbf{C}$ playing a vital role in the decoding process is called the minimum distance.

Definition 2 *The minimum distance is defined as*

$$d = min\{\mu(\boldsymbol{c}_1, \boldsymbol{c}_2) \, : \, \boldsymbol{c}_1, \boldsymbol{c}_2 \in \boldsymbol{C}, \; \boldsymbol{c}_1 \neq \boldsymbol{c}_2\}.$$

The importance of d in the decoding problem is due to the fact that any $\mathbf{x} \in \mathbb{F}_q^n$ whose distance from $\mathbf{C}$ is less than $\frac{d}{2}$, (where the distance from $\mathbf{C}$ to $\mathbf{x}$ is $\min\{\mathrm{d}(\mathbf{x}, \mathbf{c}) \, : \, \mathbf{c} \in \mathbf{C}\}$), has a unique closest codeword. For a given code length n, we therefore would like the minimum distance to be as large as possible.

1.3 The Problem Revisited

In this new language, it is relatively easy to address the problem of obtaining a low $P_{\mathbf{C}}$ value while keeping the information rate close to 1. Suppose we decide that the satellites in our example can only send 21 symbols per hour, lest their energy run out too quickly. Instead of using the repetition code of length 21, $\mathbf{C}_1$, once every hour, it may be advantageous to use a 42 symbol code, $\mathbf{C}_2$, once every two hours. In this case, $\mathbf{C}_2$ must consist of four codewords in $\mathbb{F}_2^{42}$ to account for the messages 'no no', 'no yes', 'yes no', and 'yes yes'. Indeed, it turns out that there are four word codes $\mathbf{C}_2 \subset \mathbb{F}_2^{42}$ for which $P_{\mathbf{C}_2}$ is less than the probability of incorrect message reception associated to using $\mathbf{C}_1$ twice in one hour, $P_{\mathbf{C}_1}(2 - P_{\mathbf{C}_1})$.

In fact, a basic result in coding theory called Shannon's theorem, (1948), tells us that the more information we send at one time, (for example in the above case, waiting four or five hours and more), the closer we can bring $P_{\mathbf{C}}$ to zero [1].

Nevertheless, there is still a lot of work to be done. Even though Shannon's theorem tells us that these codes have a low $P_{\mathbf{C}}$ value, there is no guarantee that there are efficient ways of decoding them. When a code lacks extra mathematical structure, the decoding procedure will amount to searching all of the codewords for the closest one to the received vector $\mathbf{x}$. This can be a very lengthy process if the code is large. As we will see, the decoding procedures can be made more efficient when there is some order to the codeword distribution in $\mathbb{F}_q^n$. We will seek out the codes among the ones mentioned in Shannon's theorem that possesses this extra structure.

1.4 The Choice of $\mathbb{F}_q$

In many applications, (satellites, telephones, computer modems, etc.), the information is being transmitted as words in only two signals, each of which can with equal probability p admit an error. Due to our assumptions in Section 1.2, we must therefore use the field $\mathbb{F}_2$ to construct codes for these cases. As a result, a large portion of the work in coding theory is focused on

[1] For a statement and proof of Shannon's theorem see [7].

codes over $\mathbb{F}_2$.

In some of these applications, codes with lower $P_{\mathbf{C}}$ value and-or quicker decoding procedures are available if we allow ourselves to use fields other than $\mathbb{F}_2$. Therefore one may be tempted to use a code over another field such as $\mathbb{F}_4$, transmitting the $\mathbb{F}_2$ vector space equivalents of the elements in $\mathbb{F}_4$. In other words if $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$, then every time we want to send the "signals" 0, 1, α, or α^2 we simply just send the pairs of signals (0,0), (0,1), (1,0), or (1,1) respectively. This effectively converts it from a code over $\mathbb{F}_4$ to a code over $\mathbb{F}_2$. If we do this, however, the resulting code over $\mathbb{F}_2$ will most likely not have the same $P_{\mathbf{C}}$ value as when the code was considered over $\mathbb{F}_4$. In addition, the ways in which the incoming messages are decoded will have to change considerably, which could result in an inefficient decoding procedure. In other words, the good qualities of the code as seen over $\mathbb{F}_4$ may not be preserved.

To see this, we take as an example the simple two word repetition code over $\mathbb{F}_4$ of length 11 and compare the decoding procedure over $\mathbb{F}_4$ with the decoding that should occur if we send these codewords through a two signal channel as described above.

Let $\mathbf{C}$ be the two word repetition code of length 11 over $\mathbb{F}_4$. Therefore

$$\mathbf{C} = \{\mathbf{0}, \mathbf{1}\} \subseteq \mathbb{F}_4^{11} \text{ where } \mathbf{0} = (0, \ldots, 0) \text{ and } \mathbf{1} = (1, \ldots, 1).$$

If $\mathbf{x} \in \mathbb{F}_4^{11}$, let $|\mathbf{x}|_1$ represents the number of ones in the entries of $\mathbf{x}$ and $|\mathbf{x}|_0$ represents the number of zeros. When receiving messages that were sent using $\mathbf{C}$, it is then clear that any received vector in the set $\{\mathbf{x} \in \mathbb{F}_4^{11} : |\mathbf{x}|_1 < |\mathbf{x}|_0\}$ should be decoded as the message $\mathbf{0}$ and any received vector in $\{\mathbf{x} \in \mathbb{F}_4^{11} : |\mathbf{x}|_1 > |\mathbf{x}|_0\}$ should be decoded as the message $\mathbf{1}$. Thus for example the incoming message

$$(\alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, 0)$$

would be decoded as $\mathbf{0}$ and the message

$$(\alpha, \alpha^2, 1, \alpha^2, \alpha, 0, \alpha^2, 1, \alpha^2, \alpha, \alpha)$$

would be decoded as $\mathbf{1}$.

Note that this decoding scheme is strongly dependent upon the assumptions laid out in Section 1.2. Let us examine the validity of these assumptions if we were to transmit the messages using only two signals '0' and '1'. First of all we notice that transmission can indeed occur with the use of four distinct "signals". These "signals" would be the pairs of signals (0,0), (0,1), (1,0), and (1,1). Next, we see that assuming that the probability of error when transmitting any of the "signals" is the same no matter what "signal" you are transmitting is also valid. In particular, since we have assumed that a 0 or a 1 can error with equal probability p, it follows that each one of (0,0), (0,1), (1,0), and (1,1) can error with equal probability

$$p' = 2p(1-p) + p^2 = p(2+p).$$

However, the last assumption is where we see a problem. That is, the assumption that transmission error will result with equal probability in any of the 3 remaining "signals". We can assume without loss of generality that we are attempting to transmit (0,0). If an error occurs, our assumption says that it is equally likely to result in the incorrect "signals" (1,0), (0,1), or (1,1). The probabilities associated to these events are easily calculated however as $p(1-p)$, $p(1-p)$, and p^2 respectively. For most applications, p is much less than a half. Therefore these values differ significantly.

This faulty assumption will have a strong impact on the effectiveness of our decoding procedure above. The problems due to "signal" dependent probabilities will magnify when considering these 11-tuple codewords. For example, we had agreed to decode the vector

$$\mathbf{x}' = (\alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, \alpha^2, 0)$$

as

$$\mathbf{0} = ((0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0)),$$

whereas the actual $\mathbb{F}_2$ transmission of $\mathbf{x}'$,

$$((1,1),(1,1),(1,1),(1,1),(1,1),(1,1),(1,1),(1,1),(1,1),(1,1),(0,0)),$$

was with much greater probability, $(p^{11}(1-p)^{11}$ as opposed to $p^{20}(1-p)^2)$, intended as the message

$$\mathbf{1} = ((0,1),(0,1),(0,1),(0,1),(0,1),(0,1),(0,1),(0,1),(0,1),(0,1),(0,1)).$$

Similarly we can find many other 11-tuples that would be decoded incorrectly if we were to follow the decoding scheme based on the assumptions of Section 1.1. Therefore we see that our transition to $\mathbb{F}_2$ requires the reevaluation of our decoding procedure. In addition, since the value of $P_{\mathbf{C}}$ is strongly dependent on the ways in which incoming vectors can be decoded, it follows that our change to $\mathbb{F}_2$ will most likely bring about a different $P_{\mathbf{C}}$ value as well.

The purpose of this section is not to drive us from the study of codes over fields other than $\mathbb{F}_2$. There are many uses to codes over these fields, even in cases where the information is being transmitted over a channel with only two signals. For example, there is a way of "combining" codes over $\mathbb{F}_2$ with codes over other fields to make error correcting schemes that can be vary effective for communicating in two signals [2]. What should be emphasized however is that codes over arbitrary fields can be applied in the case of two signal channels only after much thought and hard work. Such applications should not be taken for granted.

[2]These "combined codes" are called *concatenated codes* and are explained in detail in [3].

Chapter 2
Linear Error Correcting Codes

2.1 Definition and Properties

The subsets of $\mathbb{F}_q^n$ having the most natural structure are linear subspaces. The first step in our move towards structured codes will be to limit our attention to codes $\mathbf{C}$ for which the codeword distribution is a linear subspace of $\mathbb{F}_q^n$.

Definition 3 *A code C is a linear code of dimension k if $C = \mathcal{M} \subseteq \mathbb{F}_q^n$ where $\mathcal{M}$ is a linear subspace of $\mathbb{F}_q^n$ of dimension k.*

For the remainder of this paper, we will deal exclusively with linear codes.

It is important to note that although a linear code is defined to be nothing more than a linear subspace of $\mathbb{F}_q^n$, it is characterized by much more than its linear structure. For example, the one dimensional linear subspaces in $\mathbb{Z}_2^3$ generated by $(1,0,0)$ and $(1,1,1)$ are linearly isomorphic and therefore equivalent as linear subspaces. However, the associated two word codes $\mathcal{M}_1$ and $\mathcal{M}_2$ are far from equivalent, as their minimum distances are 1 and 3 respectively.

The number of calculations required to obtain the minimum distance for linear codes is greatly reduced from that of an arbitrary code. This is due to the fact that the metric μ is invariant under affine translations. In particular if T is an affine translation of $\mathbb{F}_q^n$ then for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{F}_q^n$, $\mu(T(\mathbf{x}_1), T(\mathbf{x}_2)) = \mu(\mathbf{x}_1, \mathbf{x}_2)$. Let $\mathcal{M}$ be a linear code. Note that for any codeword $\mathbf{c}' \in \mathcal{M}$, if T is a translation taking $\mathbf{c}'$ to $\mathbf{0}$ then

$$\{\mathbf{c} \in \mathbf{C}\} = \{T(\mathbf{c}) : \mathbf{c} \in \mathbf{C}\}.$$

Therefore due to the invariance of μ under T we have that

$$\min\{d(\mathbf{c}',\mathbf{c}) \,:\, \mathbf{c}\in\mathbf{C}\,,\ \mathbf{c}\neq\mathbf{c}'\} = \min\{d(T(\mathbf{c}'),T(\mathbf{c})) \,:\, \mathbf{c}\in\mathbf{C}\,,\ T(\mathbf{c})\neq T(\mathbf{c}')\} = \min\{d(\mathbf{0},\mathbf{c}) \,:\, \mathbf{c}\in\mathbf{C}\,,\ \mathbf{c}\neq\mathbf{0}\},$$

so that if d denotes the minimum distance for $\mathcal{M}$ then

$$d = \min\{w(\mathbf{c}) \,:\, \mathbf{c}\in\mathcal{M}, \mathbf{c}\neq 0\},$$

where $w(\mathbf{c}) = \mu(\mathbf{c},0)$ is called the *weight* of $\mathbf{c}$. Thus to obtain d for linear codes, one only needs to calculate q^k distances where k is the dimension of $\mathbf{C}$. This is a significant improvement to the $\frac{q^k(q^k-1)}{2}$ number of calculations required in the general case, and is one of the many advantages to working with linear codes.

Finally, along with the introduction of linear codes, it is useful to introduce a bilinear pairing on $\mathbb{F}_q^n$. If $\mathbf{x}$ and $\mathbf{y}$ are two vectors in $\mathbb{F}_q^n$, define

$$<\mathbf{x},\mathbf{y}> \;:=\; \sum_{i=1}^{n} x_i y_i \quad .$$

We can then define the corresponding dual code of $\mathcal{M}$ as

$$\mathcal{M}^* = \left\{\mathbf{x}\in\mathbb{F}_q^n \,:\, <\mathbf{x},\mathbf{c}> = 0 \quad \forall\, \mathbf{c}\in\mathcal{M}\right\}.$$

This dual code can be very useful when analyzing the properties of $\mathcal{M}$. In addition, taking the dual is a quick and easy way to produce new linear codes. It can turn out that $\mathcal{M}^*$, having parameters $n' = n$, $k' = n-k$, and d', is a better code than the $\mathcal{M}$ that it came from.

2.2 The Singleton Bound

In this section we discuss the problem of minimizing $P_{\mathcal{M}}$ over all possible linear codes $\mathcal{M}$ of a given length n and dimension k. Recall from Chapter 1 that a low value of $P_{\mathbf{C}}$ is only one of the three main characteristics of a good code $\mathbf{C}$.

We know that to achieve this minimum for $P_{\mathcal{M}}$, we need to maximize the value of the minimum distance d. For the purpose of identifying codes whose parameters are optimal in this sense, we search for an upper bound on the value of d for any linear code of length n and dimension k.

Theorem 1 (SingletonBound) *If $\mathcal{M}$ is a linear code with parameters (n, k, d) then*

$$d \leq n - k + 1.$$

Proof. We have that $\mathcal{M} \subseteq \mathbb{F}_q^n$. Projecting $\mathcal{M}$ onto $\mathbb{F}_q^{n-1}$ by omitting the last entry of all vectors in $\mathbb{F}_q^n$, we obtain a new code of length $n-1$. If $d > 1$, then no two of the codewords in $\mathcal{M}$ could have differed in only their last entries. Therefore if $d > 1$, this new code has the same number of codewords, q^k, and so we obtain a code in $\mathbb{F}_q^{n-1}$ whose dimension is still k and whose minimum distance is now greater than or equal to $d-1$. Repeating this process $d-1$ times gives us a code of dimension k in $\mathbb{F}_q^{n-d+1}$ with minimum distance greater than or equal to 1. In other words this final code of dimension k fits inside $\mathbb{F}_q^{n-d+1}$. Thus $k \leq n - d + 1$ and so we obtain the *Singleton bound* $d \leq n - k + 1$.

□

Therefore if a linear code $\mathcal{M}$ has parameters (n, k, d) where $d = n - k + 1$, it minimizes $P_{\mathcal{M}}$ over all other linear codes of the same length and dimension. It should be noted however that for some values of n and k, no such code can be found. Therefore for some values of n and k, the Singleton bound is not a least upper bound for d [1]. This bound will nevertheless be of great use to us.

In Chapter 3 we will discover that all rational geometric Goppa codes have minimum distances that reach the Singleton bound and therefore these codes minimize $P_{\mathcal{M}}$. Due to the range of n and k for which we can produce rational geometric Goppa codes, this will in turn prove that that if $n \leq q$, the Singleton bound is a least upper bound for all values of $k \leq n$.

For the construction of the general geometric Goppa codes in Chapter 4 we will see that the minimum distances come vary close to the Singleton bound, if not reaching it. This will provide us with proof of the fact that The fact that geometric Goppa codes in general lose little, if any, of the low $P_{\mathcal{M}}$ values seen in the rational geometric Goppa codes.

[1] For a more detailed discussion of this and other bounds on d, see [7].

Chapter 3
Rational Geometric Goppa Codes

3.1 Definition and Properties

We will now present a simple construction of a class of codes whose minimum distances meet the Singleton bound. Consider $\mathcal{P}_k = \{f \in \mathbb{F}_q[x] : \deg f \leq k-1\}$. This is a vector space of dimension k with basis $\{1, x, x^2, \ldots, x^{k-1}\}$. Let $P_1, \ldots, P_n$ be distinct elements in $\mathbb{F}_q$ and define the linear 'evaluation map'

$$ev_k : \mathcal{P}_k \to \mathbb{F}_q^n : f \mapsto (f(P_1), \ldots, f(P_n)).$$

Definition 4 *If $k \leq n$, the linear code $\mathcal{M}_k \subseteq \mathbb{F}_q^n$ is defined as*

$$\mathcal{M}_k = Im(\mathrm{ev}_k).$$

These codes are called *rational geometric Goppa codes* for reasons which will be given later. Note that since $k \leq n$ it follows that for all $f \in \mathcal{P}_k$, $ev_k(f) = 0$ if and only if $f = 0$. In other words ev_k is injective so that

$$\dim[\mathcal{M}_k] = \dim[Im(ev_k)] = \dim[\mathcal{P}_k] = k.$$

Furthermore, the elements of $\mathcal{P}_k$ cannot have more than k-1 zeros in $\mathbb{F}_q$, (and we certainly can construct elements with exactly k-1 zeros), so that the minimum distance d_k of $\mathcal{M}_k$ is given as

$$d_k = n - (k-1).$$

Therefore the minimum distance of $\mathcal{M}_k$ meets the Singleton bound and thus $P_{\mathcal{M}_k}$ is lower than any other linear code of the same length and dimension. To see more specifically how these

codes are implemented in practice we take as an example the rational geometric Goppa code over the field $\mathbb{F}_{16}$ with maximum length $n = q = 16$ and dimension $k = 12$.

Let β be a primitive element of $\mathbb{F}_{16}$. Then $\mathbb{F}_{16}$ is a vector space over $\mathbb{F}_2$ generated by 1, β, β^2, and β^3. For calculations, it is useful to have a list of the vector representation of every element in $\mathbb{F}_{16}$ corresponding to this basis. Using the minimal polynomial for β we can generate such a list.

$$\begin{array}{rclrclrclrcl}
0 & = & (0,0,0,0), & \beta^3 & = & (0,0,0,1), & \beta^7 & = & (0,1,1,1), & \beta^{11} & = & (1,1,0,1), \\
\beta^0 & = & (0,0,0,1), & \beta^4 & = & (1,0,0,1), & \beta^8 & = & (1,1,1,0), & \beta^{12} & = & (0,0,1,1), \\
\beta^1 & = & (0,0,1,0), & \beta^5 & = & (1,0,1,1), & \beta^9 & = & (0,1,0,1), & \beta^{13} & = & (0,1,1,0), \\
\beta^2 & = & (0,1,0,0), & \beta^6 & = & (1,1,1,1), & \beta^{10} & = & (1,0,1,0), & \beta^{14} & = & (1,1,0,0).
\end{array}$$

The generators of $\mathcal{M}_{12}$ as a linear subspace of $\mathbb{F}_{16}^{16}$ are simply the images of the generators of $\mathcal{P}_{12}$ under ev_k. Therefore $\mathcal{M}_{12}$ is generated by the vectors

$$\begin{array}{ll}
\xi_1 = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1), & \xi_2 = (0,1,\ldots,\beta^{14}), \\
\xi_3 = ((0)^2,(1)^2,\ldots,(\beta^{14})^2), & \ldots \\
\ldots & \xi_{12} = ((0)^{11},(1)^{11},(\beta)^{11},\ldots,(\beta^{14})^{11}),
\end{array}$$

where of course the powers of β can be reduced modulo 15.

The number of messages that this code is capable of sending is equal to 16^{12}. Therefore in practice we would represent the uncoded raw data as a 12-vector over $\mathbb{F}_{16}$, say $(o_1, o_2, \ldots, o_{12})$. When we were ready to transmit this raw data we would then "encode" it by calculating

$$\mathbf{c} = \sum_{i=1}^{12} o_i \xi_i.$$

It is during this calculation that we would utilize the list above and represent every vector entry as a linear combination of 1, β, β^2, and β^3 alone.

Finally, during transmittion there is of course a certain probability that a message will accumulate errors via some sort of noise in the communication channel. To evaluate the usefulness of this code we will compare this probability $P_{\mathcal{M}_{12}}$ with the probability that an incorrect message would be received if no code were used.

Assume that the probability of signal error is p=.001. If we did not use any code, we would simply transmit the original 12-tuple representing our raw data. In this case the probability that our transmittion is received without error is $(1-p)^{12}$. Therefore the probability of incorrect transmittion of our message is

$$P_{\text{No Code}} = 1 - (1-p)^{12} \cong 0.0119.$$

When using the code $\mathcal{M}_{12}$, our message will be received and any errors corrected as long as the number of errors is less than or equal to $\frac{d-1}{2} = \frac{q-k}{2} = 2$. Let $\mathbf{X}$ denote the random variable associated to the number of signal errors in our transmittion. Then $\mathbf{X}$ has a binomial probability distribution with mean $\mu = 16p$ and standard deviation $\sigma = \sqrt{16p(1-p)}$. Correspondingly we have that

$$P_{\mathcal{M}_{12}} = P(\mathbf{X} > 2) = P(\mathbf{X} - \mu > 2 - \mu) = P(|\mathbf{X} - \mu| > 2 - \mu),$$

where the last equality holds since $\mathbf{X} - \mu$ has probability zero of taking on any values less than $-\mu$ and $-(2-\mu) = -(2-16p) < -\mu$.

Therefore Chebyshev's inequality tells us that

$$P_{\mathcal{M}_{12}} = P\left(|\mathbf{X} - \mu| > \left(\frac{2-\mu}{\sigma}\right)\sigma\right) < \frac{1}{\left(\frac{2-\mu}{\sigma}\right)^2} < 0.0041.$$

Therefore we have decreased our chances of incorrect message reception by at least one order of magnitude while our information rate is still at a large value of $\frac{12}{16}$.

The main disadvantage of rational geometric Goppa codes is that the length n, and therefore the dimension k, is bounded above by the number of elements, q, in the finite field that you're working with. This can be problematic in practice. Shannon's theorem tells us that to achieve the small values of $P_{\mathbf{C}}$ that we want, we may need to use codes of large length. Therefore if we need to use a specific finite field, we may be stuck with a large value of $P_{\mathbf{C}}$. In the extreme case, suppose that for our purposes we are required to use the field $\mathbb{F}_2$. The most meaningful rational geometric Goppa code you can construct in this case will have parameters $(n, k, d) = (2, 1, 2)$. But for a given code with minimum distance d, we only have decoding

methods for incoming vectors whose distance from our code is strictly less than $\frac{d}{2}$. Since this code has minimum distance 1, only the codewords themselves can be "decoded". In other words this code is useless.

It should be noted that despite this disadvantage, rational geometric Goppa codes are widely regarded as some of the best codes in coding theory. In addition to their simplicity, we will see in Chapter 5 that they can be decoded quickly and without difficulty. It is for this reason that in the case were longer codes are needed, we are motivated to generalize the definition of these codes to allow for better codes rather than discarding them and starting anew. This generalization, which we carry out in the next chapter, will give us the full class of geometric Goppa codes which preserve many of the positive aspects of rational geometric Goppa codes but allow for codes of arbitrarily large length over a given finite field.

Chapter 4
Geometric Goppa Codes

4.1 Preliminaries

In this chapter we will assume familiarity with the behavior of divisors and linear systems on non-singular projective algebraic curves. Although a few definitions will be given, many terms will be used without definition, for which the reader is referred to [4].

For rational geometric Goppa codes the restriction on the length of the code comes about from a limitation on the number of elements in $\mathbb{F}_q$ to use for evaluation. For the general geometric Goppa codes this problem is simply solved. Instead of evaluating polynomials on the elements of $\mathbb{F}_q$, we generalize to the evaluation of rational functions on a selection of $\mathbb{F}_q$-rational points of a non-singular projective curve $\mathcal{C}$ over $\mathbb{F}_q$. In this way, all that is needed to produce longer codes is to find a curve with more $\mathbb{F}_q$-rational points, a process for which there are no theoretical limitations.

Of course there is the problem of which rational functions to use. If $P_1, \ldots, P_n$ are the evaluation points on $\mathcal{C}$, we will need to insure that none of the rational functions that we use have any poles at any of the P_i. To do this, we select an $\mathbb{F}_q$-rational divisor $\mathbf{G}$ on $\mathcal{C}$ with the property that $\operatorname{supp} \mathbf{G} \cap \{P_i\} = \emptyset$ for all $i = 1, \ldots, n$. We then simply use the elements from $\mathcal{L}(\mathbf{G})$ for the domain of our evaluation map where

$$\mathcal{L}(\mathbf{G}) := \{f \in \mathbb{F}_{\mathcal{C}} \ : \ f \neq 0, (f) \geq -\mathbf{G}\} \cup \{0\}$$

and (f) denotes the principal divisor associated to the element f of the function field $\mathbb{F}_{\mathcal{C}}$ associated to $\mathcal{C}$. In this way, any poles that our functions may have are contained in the

support of $\mathbf{G}$, which by assumption misses all of the P_i.

Finally, note that the class of codes resulting from the construction describing above would indeed be a generalization of the class of rational geometric Goppa codes. As the name suggests, all rational geometric Goppa codes can be constructed in this way by choosing $\mathcal{C}$ as the rational curve $\mathbb{P}^1$ and using the $\mathbb{F}_q$-rational divisor $\mathbf{G}$ equal to $k-1$ times the point at infinity.

4.2 Definition and Parameter Bounds

The definition of a geometric Goppa code, motivated by our discussions thus far, is given as follows. Let $\mathbf{D} = \mathbf{P}_1 + \ldots + \mathbf{P}_n$ be a divisor on $\mathcal{C}$ where $P_1, \ldots, P_n$ are the evaluation points mentioned above. If $\mathbf{G}$ is a divisor on $\mathcal{C}$ such that $\text{supp}\ \mathbf{G} \cap \text{supp}\ \mathbf{D} = \emptyset$, define the linear map

$$ev_{G,D} : \mathcal{L}(\mathbf{G}) \to \mathbb{F}_q^n : f \mapsto (f(P_1), \ldots, f(P_n)).$$

We have the following definition.

Definition 5 *The geometric Goppa code associated to the divisors $\boldsymbol{D}$ and $\boldsymbol{G}$, $\boldsymbol{C_{\mathcal{L}}(D, G)}$, is given as*

$$\boldsymbol{C_{\mathcal{L}}(D, G)} := \text{Im}(ev_{G,D}).$$

These codes were first introduced, (in a different form), by V.D. Goppa in 1981 [9]. [1]

We can easily express the parameters for $\mathbf{C}_{\mathcal{L}}(\mathbf{D}, \mathbf{G})$ in terms of $\mathbf{D}$ and $\mathbf{G}$. The dimension, k, will be equal to the dimension of the vector space $\frac{\mathcal{L}(\mathbf{G})}{\ker(ev_{G,D})}$. However, since $\ker(ev_{G,D}) = \mathcal{L}(\mathbf{G}-\mathbf{D})$, we have that

$$k = \dim \mathcal{L}(\mathbf{G}) - \dim \mathcal{L}(\mathbf{G} - \mathbf{D}).$$

As discussed in Chapter 2, the minimum distance for $\mathbf{C}_{\mathcal{L}}(\mathbf{D}, \mathbf{G})$, d, is equal to the minimum weight over all of it's codewords. Therefore we obtain

$$d = n - \max\{l : \dim(\mathcal{L}(\mathbf{G} - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_l})) > 0\},$$

[1] Note that in earlier work [8], V.D. Goppa introduced a different class of codes which are usually referred to as classical Goppa codes.

and since for any divisor $\mathbf{H}$, if $\deg \mathbf{H} < 0$ then $\mathcal{L}(\mathbf{H}) = \{0\}$, it follows that $d \geq n - deg(\mathbf{G})$.

For the remainder of this paper, we will assume that $\mathbf{G}$ is chosen so that $\deg \mathbf{G} < n$. In this case $\deg(\mathbf{G} - \mathbf{D}) < 0$ and again this implies that $\mathcal{L}(\mathbf{G} - \mathbf{D}) = \{0\}$. Using this, and the Riemann-Roch theorem, we obtain the final pair of bounds

$$k \geq \deg \mathbf{G} + 1 - g \text{ , and } d \geq n - \deg \mathbf{G}.$$

Putting these together, it follows that $d \geq n - k + 1 - g$. Therefore d is less than $g + 1$ away from the Singleton bound $n - k + 1$.

4.3 Code Performance

Shannon's theorem tells us that codes with arbitrarily small values of P_e exist if we are willing to consider codes with arbitrarily large lengths. As we discussed in Chapter 1, such codes would be useful provided that they have additional structure which allows them to be decoded quickly. Geometric Goppa codes are indeed rich in structure. The geometric objects with which they are constructed are proving to be useful in the construction of quick and effective decoding algorithms. Much progress has been realized in this area, and it looks as though decoding algorithms on par with the most efficient of today will be found for these new class of codes. As a result of the hopeful nature of the decoding problem, if sequences of geometric Goppa codes can be found whose P_e values converge to zero, these codes may be of great use to us. Shortly after the introduction of geometric Goppa codes, it was found that not only could sequences of geometric Goppa codes be found whose P_e values tended to zero, but that for some values of q, these sequences had P_e values which tended to zero faster than any other known code. In this section we briefly describe the processes involved in the discovery of these code sequences.

In order to construct long geometric Goppa codes, we will need to use curves $\mathcal{C}$ with a large number of rational points. On the other hand to minimize P_e for a given length, we need to maximize the minimum distance d. Therefore the bound $d \geq n - k + 1 - g$ tells us that for a given length we would like to choose the curve with as small a genus as possible. Thus if we wish to find a sequence of geometric Goppa codes whose P_e values rapidly converge to zero,

we will need to balance these needs for a large number of rational points and a small genus.

In 1983 Drinfeld and Vladut [10] succeeded in determining a bound on the relative size of n with respect to g that is attainable in the limit of large values of g. They found that if $N(g)$ denotes the maximum number of rational points that can exists on a curve of genus g over $\mathbb{F}_q$ then

$$\limsup_{g\to\infty} \frac{N(g)}{g} \leq \sqrt{q} - 1.$$

Therefore we know that the the lowest genus we can achieve in the limit of large n would be realized in sequences of curves whose values of $\frac{N(g)}{g}$ converge to this limit.

In 1982 Tsfasman, Vladut, and Zink [6] used modular curves to construct a sequence of curves converging to the Drinfeld-Vladut bound for the case where q is a square. For $q \geq 49$ they showed that the codes arising from these curves have P_C values which converge to zero quicker than any previously known code. Their constructions however were not simple and explicit enough to be used for the construction of actual codes.

Finally, in 1995 Garcia and Stichtenoth [1] were able to construct, (simply and more explicitly), a sequence of curves converging to the Drinfeld-Vladut bound. Their construction was again done for the case where q was a square only. Although they did not construct any codes from these curves, it looks hopeful that such constructions will follow.

It is important to note that although it is taking time to construct explicit codes demonstrating the fact that geometric Goppa codes can most rapidly approach small values of P_C for large n, there are many geometric Goppa codes that have been explicitly constructed that posses parameter values on par with the best codes of this day.

We still haven't addressed the question of whether or not enough structure exists in these codes to allow for decoding algorithms which are as efficient as decoding algorithms known for other codes. Recent advances in the search for such algorithms suggests that they will be found. A survey of these advances can be found in [5]. In the next chapter, we will present one of the landmark achievements in this effort due to A.N. Skorobogatov and S.G. Vladut.

Note that when working with geometric Goppa codes, it will be useful to relax the notation for the bilinear pairing $<,>$ on $\mathbb{F}_q^n$.

Definition 6 *Suppose f is a function whose poles miss the support of $\boldsymbol{D}$. If $\mathbf{x} \in \mathbb{F}_q^n$, we then define*

$$\langle f, \mathbf{x} \rangle \ := \ \langle \mathbf{y}, \mathbf{x} \rangle \ , \ \textit{where } \mathbf{y} = (f(P_1), \ldots , f(P_n)).$$

Chapter 5
Decoding Algorithms for Geometric Goppa Codes

5.1 Basic Algorithm

Let $\mathbf{C}$ be some geometric Goppa code defined on a curve $\mathcal{C} \subseteq \mathbb{P}^2$, and having parameter values n, k and d. Our goal in this section is to outline an algorithm for decoding any vector $\mathbf{a}$ whose distance from $\mathbf{C}$ is less than or equal to t, where t is an integer with $t < \frac{d}{2}$. As discussed in the first chapter, the objective in decoding $\mathbf{a}$ is to find a unique codeword, $\mathbf{c}$, whose entries deviate from $\mathbf{a}$'s in the least number of positions. We shall call such a $\mathbf{c}$ the codeword associated to $\mathbf{a}$. Equivalently we could search for a unique n-tuple $\mathbf{e}$, having minimal weight, for which $\mathbf{a} - \mathbf{e}$ is a codeword. Such an $\mathbf{e}$ shall be called the error vector associated to $\mathbf{a}$.

Let $m_1^*, \ldots, m_{n-k}^*$ be a basis for the dual code $\mathbf{C}^*$. Suppose a vector $\mathbf{x}'$ has the property that $\mathbf{a} - \mathbf{x}' = \mathbf{c} \in \mathcal{M}$. Then for all $i = 1, \ldots, n-k$, we have $< \mathbf{a}, m_i^* > = < \mathbf{c} + \mathbf{x}', m_i^* > = < \mathbf{x}', m_i^* >$.

Definition 7 *The syndrome of* $\mathbf{a}$ *is given as*

$$s(\mathbf{a}) := (< \mathbf{a}, m_1^* >, \ldots, < \mathbf{a}, m_{n-k}^* >).$$

Then $\mathbf{x}'$ satisfies the system of equations:

$$< \mathbf{x}, m_i^* > = s(\mathbf{a})_i \quad \forall i = 1, \ldots, n-k.$$

Conversely, if $\mathbf{x}'$ is a solution to 5.1, then $< \mathbf{a} - \mathbf{x}', m_i^* > = 0$ for all i, so that $\mathbf{a} - \mathbf{x}'$ is an element of $(\mathbf{C}^*)^* = \mathbf{C}$. In other words, $\mathbf{c} = \mathbf{a} - \mathbf{x}'$ is a codeword. Therefore $\mathbf{a} - \mathbf{x}' \in \mathbf{C}$ if and

only if $\mathbf{x}'$ is a solution to 5.1. Thus finding a unique minimal weight solution to 5.1 is our goal in decoding $\mathbf{a}$.

The unique minimal weight solution to 5.1 could certainly be found by searching the finite number of solution vectors $\mathbf{x}$ for the one with minimal weight. This is as time consuming, however, as searching all of the codewords in $\mathbf{C}$ for the one closest to $\mathbf{a}$, which is exactly the task we are trying to quicken. There are more timely methods for finding the minimal weight solution to 5.1, which involve predetermining a number of zero entries for $\mathbf{x}$. The goal is to find enough zero entries in $\mathbf{x}$ so that since it further solves 5.1, it must be unique and of minimal weight.

For the cases discussed in this paper, the predetermination of zero entries in $\mathbf{x}$ will take the amount of time needed to solve another system of linear equations such as 5.1. The idea is to find what is called an 'error locator function' for $\mathbf{a}$.

Let $I = \{P_{i_1}, \ldots, P_{i_s}\}$ be the points on $\mathcal{C}$ corresponding to the set of error locations for $\mathbf{a}$. In other words, $e_j \neq 0$ if and only if $j \in \{i_1, \ldots, i_s\}$, (note that $|I| \leq t < \frac{d}{2}$).

Definition 8 *Let f be a function that is defined and not identically zero on the evaluation points $P_1, \ldots, P_n$. If $J = \{P_i : f(P_i) = 0\}$ then f is an error locator function if $I \subseteq J$ and $|J| < d$.*

If such a function is found, we will know that the vector solution to 5.1 is zero in all of the entries i for which $f(P_i)$ is not zero. We will therefore preset $x_i = 0$ for all P_i not in J. It is important to note that although $I \subseteq J$, in general $I \neq J$. Therefore f will not tell us every zero of $\mathbf{e}$. However, since $|J| < d$, we will have a sufficient number of predetermined zeros. To see this, suppose that $\mathbf{e}_1$ and $\mathbf{e}_2$ are two solutions of 5.1 with $(\mathbf{e}_1)_i = (\mathbf{e}_2)_i = 0$ for all i such that P_i is not in J. Let $\mathbf{c}_1 = \mathbf{a} - \mathbf{e}_1$ and $\mathbf{c}_2 = \mathbf{a} - \mathbf{e}_2$ be the corresponding codewords. Then $w(\mathbf{c}_1 - \mathbf{c}_2) = w(\mathbf{e}_1 - \mathbf{e}_2) < d$ so from the definition of minimum distance, it follows that $\mathbf{c}_1 = \mathbf{c}_2$. Thus $\mathbf{e}_1 = \mathbf{e}_2$ is a unique solution and is subsequently of minimal weight.

Therefore, our outline for decoding a vector $\mathbf{a}$ whose distance from $\mathbf{C}$ is less than or equal to t is as follows.

Basic Algorithm for decoding up to $t < \frac{d}{2}$ errors for geometric Goppa codes

(i) Obtain an error locator function, f, of $\mathbf{a}$.

(ii) Solve system 5.1 with the extra conditions: $x_i = 0$ for all P_i not in J.

(iii) The solution, $\mathbf{e}$, in (ii) will be a unique minimal weight solution as we saw above. Set $\mathbf{c} = \mathbf{a} - \mathbf{e}$, then $\mathbf{c}$ is the codeword associated to $\mathbf{a}$.

The specific means by which we will find the error locator function will be discussed in the following sections.

5.2 Decoding Rational Geometric Goppa Codes

5.2.1 Algorithm

We will now see that the decoding problem for rational geometric Goppa codes is easily solved. Not only are the algorithms easily understood, but they will effectively decode a large number of errors. In particular, the algorithm constructed in this section will decoding any n-tuple, $\mathbf{a}$, whose distance from the code is strictly less than $\frac{d}{2}$, where d is the minimum distance for the code. In other words, in the notation of Section 5.1, we shall set $t = [|\frac{d-1}{2}|]$.

Let $\mathbf{C} = \mathcal{M}_k$ be a rational geometric Goppa code over $\mathbb{F}_q$, as defined in Chapter 3. Let the length, n, be maximal so that $\{P_1, \ldots, P_q\} = \mathbb{F}_q$. Thus $\mathbf{C} = \mathcal{M}_k$ has parameter values $n = q$, k, and $d = q + 1 - k$.

It will be useful to view $\mathbf{C}$ as the dual of another rational geometric Goppa code. As with any linear code, we have that $[\mathbf{C}^*]^* = \mathbf{C}$. Therefore, it would suffice to express $\mathbf{C}^*$ as a rational code. Indeed it can be shown, (see Chapter II of [4]), that $\mathbf{C}^*$ is expressible as another rational geometric Goppa code, $\mathcal{M}_{k'}$, with parameter values $n' = q$, $k' = q - k$, and $d' = k + 1$.

Let $\mathbf{a}$ be an n-tuple whose distance from $\mathbf{C}$ is less than or equal to t, were t is given above. Following Section 5.1, we need to find an error locator function for $\mathbf{a}$. Let $\mathbf{e}$ be the error vector associated to $\mathbf{a}$. Since the number of non-zero elements in $\mathbf{e}$ does not exceed t, there will exist polynomials in $\mathcal{P}_{t+1}$ whose zeros contain the set of error locations, I. Since the number of zeros for any polynomial in $\mathcal{P}_{t+1}$ cannot exceed t, and $t < d$, it follows that any one of these polynomials will be an error locator function for $\mathbf{a}$. In fact, these error locator functions can be found by solving a system of linear equations as follows.

Claim 1 *A polynomial* $f = \sum_{\mu=1}^{t+1} \alpha_\mu z^{\mu-1} \in \mathcal{P}_{t+1}$ *is an error locator function for* $\mathbf{a}$ *if and only if* $(x_1, \ldots, x_{t+1}) = (\alpha_1, \ldots, \alpha_{t+1})$ *satisfies the linear system*

$$\sum_{\mu=1}^{t+1} \left(< z^{\mu-1} z^\nu, \mathbf{a} > x_\mu \right) = 0 \quad \forall \nu = 0, \ldots, k' - t - 1. \tag{5.1}$$

Proof. Suppose that $f = \sum_{\mu=1}^{t+1} \alpha_\mu z^{\mu-1} \in \mathcal{P}_{t+1}$ is an error locator polynomial. Then for all $\nu = 0, \ldots, k' - t - 1$ we have

$$\begin{aligned} \textstyle\sum_{\mu=1}^{t+1} \left(< z^{\mu-1} z^\nu, \mathbf{a} > \alpha_\mu \right) &= < \left(\textstyle\sum_{\mu=0}^{t} \alpha_\mu z^{\mu-1} \right) z^\nu, \mathbf{a} > = \langle f z^\nu, \mathbf{a} \rangle = \langle f z^\nu, \mathbf{e} + \mathbf{c} \rangle \\ &= \langle f z^\nu, \mathbf{e} \rangle = \textstyle\sum_{i=1}^{q} f(P_i)(P_i)^\nu e_i = 0 \end{aligned}$$

where the $\mathbf{c}$ term disappears since $f z^\nu \in \mathcal{P}_{k'}$ is one of the defining functions for $\mathbf{C}^*$, and the last equation holds from the definition of error locator functions. Therefore we have that $(x_1, \ldots, x_{t+1}) = (\alpha_1, \ldots, \alpha_{t+1})$ is a solution to System 5.2.

Conversely, suppose System 5.2 holds for $(x_1, \ldots, x_{t+1}) = (\alpha_1, \ldots, \alpha_{t+1})$ and $f = \sum_{\mu=1}^{t+1} \alpha_\mu z^{\mu-1}$. Furthermore, suppose there were a $P_{i'} \in I$ with $f(P_{i'}) \neq 0$. Let $g = \sum_{\nu=0}^{k'-t-1} \beta_\nu z^\nu \in \mathcal{P}_{k'-t}$ be such that g is zero on all of I except $P_{i'}$, (certainly such a g exists since its required zeros number $|I| - 1 \leq t - 1 \leq d - t - 2 = k' - t - 1$). Then from 5.2 we have

$$0 = \sum_{\nu=0}^{k'-t-1} \left(\beta_\nu \langle f z^\nu, \mathbf{a} \rangle \right) = \langle fg, \mathbf{a} \rangle = \langle fg, \mathbf{e} \rangle = \sum_{i=1}^{q} f(P_i) g(P_i) e_i = f(P_{i'}) g(P_{i'}) e_{i'},$$

which is a contradiction since all of $f(P_{i'})$, $g(P_{i'})$, and $e_{i'}$ are non-zero. Thus f must be an error locator function which proves the claim.

□

Therefore, we have the following algorithm:

Decoding up to $t \leq [|\frac{d-1}{2}|]$ errors for rational geometric Goppa codes

(i) Find a non-trivial solution, $(\alpha_1, \ldots, \alpha_{t+1})$, to the linear system 5.2, obtaining the error locator function $f = \sum_{i=1}^{t+1} \alpha_i z^{i-1}$.

(ii) Solve the linear system 5.1 for $\mathbf{x}$, with the extra conditions: $x_i = 0$ for all i not in J, (where J is the zero set of f in $\mathbb{F}_q$).

(iii) Set $\mathbf{e} = \mathbf{x}$, the decoded word is then $\mathbf{c} = \mathbf{a} - \mathbf{e}$.

5.2.2 'C'-version of algorithm

As a demonstration of the results in Subsection 5.2.1, I have written a program implementation of the algorithm, written in C, for an arbitrary prime field $\mathbb{F}_p$. The program can be found on my web site at `www.math.ubc.ca`. It uses row reduction to solve the matrix equations associated to the linear systems 5.1 and 5.2. The program procedure is as follows:

1) Input an n-tuple $\mathbf{a}$.

2) Find a non-trivial solution to the matrix equation:

$$\begin{bmatrix} - & & - \\ - & \gamma_1 & - \\ - & \gamma_2 & - \\ & \vdots & \\ - & \gamma_{q-k-t} & - \end{bmatrix} \begin{bmatrix} | \\ \alpha \\ | \end{bmatrix} = \begin{bmatrix} | \\ 0 \\ | \end{bmatrix}$$

where $\gamma_1, \ldots, \gamma_{q-k-t}$ are given by

$$\begin{bmatrix} - & \gamma_j & - \end{bmatrix} = \begin{bmatrix} a_1 P_1^{j-1} & \ldots & a_q P_q^{j-1} \end{bmatrix} \begin{bmatrix} 1 & P_1 & \ldots & P_1^t \\ 1 & P_2 & \ldots & P_2^t \\ & & \vdots & \\ 1 & P_q & \ldots & P_q^t \end{bmatrix}$$

3) Set $f(z) = \sum_{i=0}^{t} \alpha_i z^i$ and determine J, (the zero set of f in $\mathbb{F}_p$).

4) Find the unique solution, $\mathbf{x}$, to the system of equations

$$\begin{bmatrix} 1 & \dots & 1 \\ P_1 & \dots & P_q \\ & \vdots & \\ P_1^k & \dots & P_q^k \end{bmatrix} \begin{bmatrix} | \\ \overline{\mathbf{x}} \\ | \end{bmatrix} = \begin{bmatrix} | \\ \mathbf{b} \\ | \end{bmatrix}$$

where $\overline{\mathbf{x}}$ and $\mathbf{b}$ are given as

$$b_j = \sum_{i=1}^{q} a_i P_i^{j-1}, \quad \text{and} \quad \overline{x_j} = \begin{cases} x_j & \text{if } j \in J \\ 0 & \text{otherwise} \end{cases}$$

Finally, if either one of steps 2) or 4) cannot be performed, then the algorithm should terminate with the knowledge that $\mathbf{a}$ has more than t errors and therefore cannot be decoded.

5.2.3 Complexity

As we have noted, the most basic decoding algorithm is performed by first computing the distances $\mu(\mathbf{a}, \mathbf{c})$ for all of the codewords $\mathbf{c} \in \mathbf{C}$ and subsequently decoding $\mathbf{a}$ as the codeword $\mathbf{c}$ that produces the smallest distance.

To demonstrate the improved speed with which we can decode incoming vectors when using the algorithm presented in this section, we now compare the number of bit operations required in the algorithm of Subsection 5.2.1 with the number required in the basic decoding algorithm.

Let $\mathcal{M}_k \subseteq \mathbb{F}_q^p$ be a rational geometric Goppa code of length p over $\mathbb{F}_p$ for some prime p. We will assume that the information rate of $\mathcal{M}_k$ is greater than $\frac{1}{2}$ so that $k > \frac{p}{2}$. Let $\mathbf{a} \in \mathbb{F}_p^p$ be the vector that we wish to decode with $\mathrm{d}(\mathbf{a}, \mathbf{C}) \leq \frac{d-1}{2}$ where d is the minimum distance of $\mathcal{M}_k$ so that indeed $\mathbf{a}$ can be decoded.

For each of the decoding algorithms, we will estimate the number of additions-subtractions and multiplications-divisions that are required to decode $\mathbf{a}$, (we will count reductions modulo p as one division). Assuming an average bit length of $N = \frac{1}{2}\log_2 p$ for every number that is being

operated on, we can then denote the number of bit operations required to perform one addition or subtraction as N. For simplicity we will assume that the slowest multiplication and division procedures are being used so that the number of bit operations for a multiplication or division will be N^2. In this way, we will finally estimate the number of bit operations required in each algorithm and compare.

In the case of the basic decoding algorithm we need to calculate one distance for every codeword in $\mathcal{M}_k$ and subsequently compare that distance to the smallest distance that we had computed up to that point. Each distance will involve p subtractions and p additions and the comparison will involve one subtraction. Therefore since this is done for all of the p^k codewords, we can estimate the total number of bit operations as $p^k(2p+1)N > p^{p/2}(2p+1)N$ so that

$$\text{Number of bit operations required in basic algorithm} \geq O\left((\log_2 p)(p^{\frac{p+2}{2}})\right).$$

For the algorithm of this section, there are five main steps that require the bulk of the computing time. These are

(i) The computations required to set up the matrix associated to the first system of linear equations.

(ii) The gaussian elimination needed to find the solution to this first system.

(iii) The calculations required to determine the zero set of the error locator function.

(iv) The computations required to set up the matrix associated to the second system of linear equations.

(v) The gaussian elimination needed to find the solution to this second system.

Using the information in Subsection 5.2.2, and the fact that gaussian elimination on an m by n matrix would require $O(m^2n)$ multiplications and $O(mn)$ additions, we can estimate that the order with respect to the variables p,k, and t of addition-subtractions and multiplications-divisions are respectively:

(i): $O(p^2t - pkt - pt^2)$ and $O(p^3t - p^2kt + p^2t^2 - pk^2t - pkt^2 - pt^3)$,

(ii): $O(pt - kt - t^2)$ and $O(p^2t - pkt - pt^2 + kt^2 + t^3)$,

(iii): $O(pt)$ and $O(t^2)$,

(iv): $O(p^2)$ and $O(pk^2 + p^3)$,

(v): $O(pk)$ and $O(pk^2)$.

However, since $\frac{q}{2} < k < q$ and $0 \leq t \leq \frac{q-k}{2}$, it follows that $k = O(q)$ and $t \leq O(q)$. Therefore we can obtain upper bounds on the order of the expressions above. We then have:

(i): $O(p^3)$ and $O(p^4)$,

(ii): $O(p^2)$ and $O(p^3)$,

(iii): $O(p^2)$ and $O(p^2)$,

(iv): $O(p^2)$ and $O(p^3)$,

(v): $O(p^2)$ and $O(p^3)$.

Adding these together we see that in the decoding algorithm of this section, we require at most $O(p^3)$ additions-subtractions and $O(p^4)$ multiplications-divisions. From this we conclude:

Number of bit operations required for the algorithm in this section $\leq O\left((\log_2 p)^2 p^4\right)$.

We see therefore that the algorithm in this section is indeed an improvement on the basic decoding algorithm, as it competes in polynomial time. Although for small values of p, this improvement we not be seen, the increased speed for large values of p is tremendous. We are particularly concerned with improved speed for large values of p since one of our main concerns is to endow the longer codes of Shannon's theorem with quick decoders and for rational geometric Goppa codes p is large if we are considering long codes.

5.3 Decoding Geometric Goppa Codes

Let $\mathbf{C} = \mathbf{C}_{\mathcal{L}}(\mathbf{D}, \mathbf{G})$ be a geometric Goppa code defined for the rational divisors $\mathbf{D}$ and $\mathbf{G}$ on a non-singular projective curve $\mathcal{C}$ over $\mathbb{F}_q$. In this section, we will derive a decoding algorithm for $\mathbf{C}$ due to A.N. Skorobogatov and S.G. Vladut [2], following an idea of J. Justesen et al.

As in Section 5.2, we will rely heavily on the dual of $\mathbf{C}$ which is again an algebraic geometric code. In particular, $\mathbf{C}^* = \mathbf{C}_{\mathcal{L}}(\mathbf{D}, \mathbf{H})$, where $\mathbf{H}$ is another rational divisors on $\mathcal{C}$ with the property that $\deg \mathbf{H} = \deg \mathbf{D} - \deg \mathbf{G} + 2g - 2$, (see Chapter II of [4]).

Unlike the case for rational geometric Goppa codes, the algorithm in this section will not necessarily be able to decode all vectors whose distance from $\mathbf{C}$ is less than half the minimum distance. To date, the best decoding algorithms for geometric Goppa codes can only guarantee the decoding of less than or equal to $\frac{d^*-1}{2}$ errors, where $d^* = n - \deg \mathbf{G}$ is the design distance of $\mathbf{C}$, (see [5]). The algorithm in this section will fall somewhat short of that mark, with a guaranteed correction of any error vector having weight less than or equal to $\frac{d^*-1-g}{2}$ where g is the genus of $\mathcal{C}$. Therefore the algorithm in this section will decode the maximal possible errors, $\frac{d-1}{2}$, in the case that $\mathbf{C} = \mathbb{P}^1$ where $g = 0$. In fact, the algorithm in Section 5.2 is the application of this section to $\mathcal{C} = \mathbb{P}^1$.

Let $\mathbf{a}$ be the vector that we wish to decode. As in Section 5.2, we would like to find a function space within which the error locator function of $\mathbf{a}$ can be determined by solving a system of linear equations. Let t be an integer with $0 < t < \frac{d^*-1}{2}$ and assume that the distance from $\mathbf{a}$ to $\mathbf{C}$ is less than or equal to t. The following theorem will give conditions under which we can find such an appropriate function space.

Theorem 2 *If there exists a divisor $\boldsymbol{G}_1$ on $\mathcal{C}$ such that*

$$\begin{cases} \text{supp } \boldsymbol{G}_1 \cap \text{supp } \boldsymbol{D} = \emptyset, \\ \deg \boldsymbol{G}_1 < d^* - t, \text{ and} \\ \dim \mathcal{L}(\boldsymbol{G}_1) > t, \end{cases} \tag{5.2}$$

then $\mathcal{L}(\boldsymbol{G}_1)$ contains an error locator function for $\mathbf{a}$. In particular, $f \in \mathcal{L}(\boldsymbol{G}_1)$ is an error

locator function if and only if $f = \sum_{\mu=1}^{l_1} \alpha_\mu \xi_\mu$ *where* $(x_1, \ldots, x_{l_1}) = (\alpha_1, \ldots, \alpha_{l_1})$ *is a non-trivial solution to the linear system*

$$\sum_{\mu=1}^{l_1} (\langle \xi_\mu \eta_\nu, \mathbf{a} \rangle \, x_\mu) = 0 \quad \forall \nu = 1, \ldots, l_2, \tag{5.3}$$

where $\{\xi_1, \ldots, \xi_{l_1}\}$ *and* $\{\eta_1, \ldots, \eta_{l_2}\}$ *are basis of the spaces* $\mathcal{L}(\boldsymbol{G}_1)$ *and* $\mathcal{L}(\boldsymbol{H} - \boldsymbol{G}_1)$ *respectively.*

Proof. Suppose that $I = \{P_{i_1}, \ldots, P_{i_s}\}$ is the set of error locations on $\mathbf{D}$, (note that $s \leq t$). We would first like to show that there exists a non-zero function in $\mathcal{L}(\mathbf{G}_1)$ whose zero set contains all of I. Since the supports of $\mathbf{G}_1$ and $\mathbf{D}$ are disjoint, such functions will exist if and only if the linear system $\mathcal{L}(\mathbf{G}_1 - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_s})$ is non-trivial. But $\mathbf{G}_1 - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_s} < \mathbf{G}_1$, so that

$$\mathcal{L}(\mathbf{G}_1 - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_s}) \subseteq \mathcal{L}(\mathbf{G}_1).$$

Therefore we have that

$$\dim \mathcal{L}(\mathbf{G}_1) - \dim \mathcal{L}(\mathbf{G}_1 - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_s}) \leq \deg \mathbf{G}_1 - \deg(\mathbf{G}_1 - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_s}) = s,$$

and since $\dim \mathcal{L}(\mathbf{G}_1) \geq t + 1$, it follows that

$$\dim \mathcal{L}(\mathbf{G}_1 - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_s}) \geq t - s + 1 \geq 1.$$

Hence $\mathcal{L}(\mathbf{G}_1)$ does indeed contain non-zero functions whose zero sets contain I. In fact, these functions are error locator functions. To see this, suppose f is such a function and $J = \{P_{j_1}, \ldots, P_{j_r}\}$ is it's corresponding set of zeros on $\mathbf{D}$. It then follows that $\mathcal{L}(\mathbf{G}_1 - \mathbf{P}_{j_1} - \ldots - \mathbf{P}_{j_r}) \neq \{0\}$. If it were the case that $|J| = r > d^*$, then

$$\deg(\mathbf{G}_1 - \mathbf{P}_{j_1} - \ldots - \mathbf{P}_{j_r}) = \deg \mathbf{G}_1 - r < -t < 0$$

which is a contradiction. Therefore $|J| < d^* \leq d$ so that f is in fact an error locator function for $\mathbf{a}$.

Now suppose that $f \in \mathcal{L}(\mathbf{G}_1)$ is an error locator function. Since $\{\xi_\mu\}$ is a basis for $\mathcal{L}(\mathbf{G}_1)$, we can find an $(\alpha_1, \ldots, \alpha_{l_1}) \in \mathbb{F}_q^{l_1}$ such that $f = \sum_{\mu=1}^{l_1} \alpha_\mu \xi_\mu$. Note in addition that $f\eta_\nu \in \mathcal{L}(\mathbf{H})$

for all ν, and is therefore an element of the generating class of functions for $\mathbf{C}^*$. Therefore for all ν we have

$$\sum_{\mu=1}^{l_1} (\langle \xi_\mu \eta_\nu, \mathbf{a} \rangle \alpha_\mu) = \sum_{\mu=1}^{l_1} \langle \alpha_\mu \xi_\mu \eta_\nu, \mathbf{a} \rangle = \langle f\eta_\nu, \mathbf{c}+\mathbf{e} \rangle = \langle f\eta_\nu, \mathbf{e} \rangle = \sum_{i=1}^{n} f(P_i)\eta(P_i)e_i = 0,$$

since f is an error locator function, and thus system 5.4 holds for $(x_1, \ldots, x_{l_1}) = (\alpha_1, \ldots, \alpha_{l_1})$.

Conversely suppose that $f = \sum_{\mu=1}^{l_1} \alpha_\mu \xi_\mu \in \mathcal{L}(\mathbf{G}_1)$ for some solution $(\alpha_1, \ldots, \alpha_{l_1})$ to the system 5.4. Assume that there is a $P_{i'} \in I$ with $f(P_{i'}) \neq 0$. As in Section 5.2, we will obtain a contradiction through the use of a function $h \in \mathcal{L}(\mathbf{H} - \mathbf{G}_1)$ that is zero on all of I except $P_{i'}$. To show that such an h exists, we calculate

$$\begin{aligned} \deg(\mathbf{H} - \mathbf{G}_1 - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_s}) &= \deg(\mathbf{D}) - \deg(\mathbf{G}) + 2g - 2 - \deg(\mathbf{G}_1) - s \\ &> d^* + 2g - 2 - (d^* - t) - t = 2g - 2. \end{aligned}$$

Therefore, using the Riemann-Roch theorem to calculate the dimensions of the respective vector spaces, we can then verify the proper inclusion

$$\mathbf{V}_1 := \mathcal{L}(\mathbf{H} - \mathbf{G}_1 - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_s}) \subset \mathcal{L}(\mathbf{H} - \mathbf{G}_1 - \mathbf{P}_{i_1} - \ldots - \mathbf{P}_{i_s} + \mathbf{P}_{i'}) =: \mathbf{V}_2.$$

Hence, there exists a non-zero $h \in \mathbf{V}_2 - \mathbf{V}_1 \subseteq \mathcal{L}(\mathbf{H} - \mathbf{G}_1)$. This h has the desired properties. From 5.4 we then have

$$0 = \langle fh, \mathbf{a} \rangle = \langle fh, \mathbf{e} \rangle = \sum_{i=1}^{n} f(P_i)h(P_i)e_i = f(P_{i'})h(P_{i'})e_{i'}$$

which is a contradiction since all of $f(P_{i'})$, $h(P_{i'})$, and $e_{i'}$ are non-zero. Therefore if J is the zero set of f in $\{P_1, \ldots, P_n\}$, then $I \subseteq J$ and since $f \in \mathcal{L}(\mathbf{G}_1)$ it follows that $|J| \leq deg(G_1) < d^* \leq d$. Hence f is an error locator function for $\mathbf{a}$.

□

Thus if we have a divisor $\mathbf{G}_1$ satisfying 5.3, we can use it to decode n-tuples, $\mathbf{a}$, whose distances from $\mathbf{C}$ is less than or equal to t. All that remains is to determine values of t for which such a divisor can be found. The following theorem does exactly this.

Theorem 3 *If* $0 \leq t \leq \frac{d^*-1-g}{2}$, *there exists a divisor* $\boldsymbol{G}_1$ *on* $\mathcal{C}$ *satisfying 5.3.*

Proof. See Chapter VII of [4].

□

Once we obtain the $\mathbf{G}_1$ proven to exist in Theorem 3, we have the following algorithm.

Decoding up to $t \leq \frac{d^*-1-g}{2}$ errors for a geometric Goppa code $\mathbf{C}_{\mathcal{L}}(\mathbf{D}, \mathbf{G})$

(i) Find a non-trivial solution, $(\alpha_1, \ldots, \alpha_{l_1})$, to the linear system 5.4, obtaining the error locator function $f = \sum_{\mu=1}^{l_1} \alpha_\mu \xi_\mu$.

(ii) Solve the linear system 5.1 for $\mathbf{x}$, with the extra conditions: $x_i = 0$ for all i not in J, (where J is the zero set of f in $\mathbb{F}_q$).

(iii) Set $\mathbf{e} = \mathbf{x}$. The codeword associated to $\mathbf{a}$ is then $\mathbf{c} = \mathbf{a} - \mathbf{e}$.

As with the decoding algorithm in Section 5.2, the majority of the computation in this algorithm involves finding solutions to two main systems of linear equations. The time required to decode a vector will therefore be polynomial in q. Hence as in Section 5.2, for large values of q the algorithm in this section will help us to decode vectors much quicker than by using the basic decoding algorithm.

Bibliography

[1] Garcia A. and Stichtenoth H. A tower of artin-schreier extensions of function fields attaining the drinfeld-vladut bound. *Invent. Math.*, 121:211–222, 1995.

[2] Skorobogatov A.N. and Vladut S.G. On decoding albebraic-geometric codes. *IEEE Trans. Inform. Th.*, 36:1461–1463, 1990.

[3] Jr. Forney D. G. *Concatenated Codes*, volume 37 of *Research Monographs*. The MIT Press, 1966.

[4] Stichtenoth H. *Algebraic Function Fields and Codes*. Universitext. Springer-Verlag, 1993.

[5] Høholdt T. and Pellikaan R. On the decoding of algebraic-geometric codes. *IEEE Trans. Inform. Th.*, 41(6), November 1995.

[6] Vladut S.G. Tsfasman M.A. and Zink T. On goppa codes which are better than the gilbert-varshamov bound. *Math. Nachr.*, 109:21–28, 1982.

[7] van Lint J.H. *Introduction to Coding Theory*, volume 86 of *Graduate Texts in Mathematics*. Springer-Verlag, 1982.

[8] Goppa V.D. A new class of linear error-correcting codes. *Problems of Info. Transmition*, 6:207–212, 1970.

[9] Goppa V.D. Codes on algebraic curves. *Soviet Math. Dokl.*, 24(1):170–172, 1981.

[10] Drinfeld V.G. and Vladut S.G. Number of points of an algebraic curve. *Func. Anal.*, 17:53–54, 1983.