

To the Graduate Council:

I am submitting herewith a dissertation written by Justin Groves entitled “An Exploration of Absolute Minimal Degree Lifts of Hyperelliptic Curves.” I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mathematics.

Luís Finotti, Major Professor

We have read this dissertation
and recommend its acceptance:

Michael W. Berry

Ioannis Sgouralis

Shashikant Mulay

Marie Jameson

Accepted for the Council:

Dixie Thompson

Vice Provost and Dean of the Graduate School

To the Graduate Council:

I am submitting herewith a dissertation written by Justin Groves entitled “An Exploration of Absolute Minimal Degree Lifts of Hyperelliptic Curves.” I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mathematics.

Luís Finotti, Major Professor

We have read this dissertation
and recommend its acceptance:

Michael W. Berry

Ioannis Sgouralis

Shashikant Mulay

Marie Jameson

Accepted for the Council:

Dixie Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

An Exploration of Absolute Minimal Degree Lifts of Hyperelliptic Curves

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Justin Groves

August 2023

© by Justin Groves, 2023
All Rights Reserved.

For my God my Redeemer, and my wife, Amberly, my rescuer.

Acknowledgements

I would like to thank my advisor, Luís Finotti, whose help and expertise were invaluable throughout my graduate school experience, and the Department of Mathematics faculty who served as coordinators for the courses I taught. Their feedback and advice have made me into the teacher that I am at heart today. Special thanks also goes to Michael Berry, Marie Jameson, Shashikant Mulay, and Ioannis Sgouralis for serving on my doctoral committee.

I would also like to thank Ian and Ginny Neal, along with the COGWA Knoxville church congregation for their unwavering emotional and spiritual support during this journey. Discussions with my office mate Jeremiah Smith were also enlightening and supportive. Finally, I would like to thank all my friends and family (too many to list) for their additional encouragement.

God made the integers; all else is the work of man. - Leopold Kronecker

Pure mathematics is, in its way, the poetry of logical ideas. - Albert Einstein

Abstract

For any ordinary elliptic curve E over a field with non-zero characteristic p , there exists an elliptic curve $\mathbf{E}$ over the ring of Witt vectors $\mathbf{W}(\mathbf{E})$ for which we can lift the Frobenius morphism, called the canonical lift. Voloch and Walker used this theory of canonical liftings of elliptic curves over Witt vectors of length 2 to construct non-linear error-correcting codes for characteristic two. Finotti later proved that for longer lengths of Witt vectors there are better lifts than the canonical. He then proved that, more generally, for hyperelliptic curves one can construct a lifting over the ring of Witt vectors with minimal degrees of the coordinate polynomials of the lift of points, and that these achieve better parameters for the resulting non-linear codes. In this dissertation we describe an algorithm to compute these absolute minimal degree liftings of hyperelliptic curves, along with constructing non-linear codes over finite fields of non-zero characteristic greater than two. From our algorithm and exploratory analysis we derive results and conjectures regarding requirements to compute absolute minimal degree lifts satisfying the theoretical lower bound, i.e., the ones that give the best bounds for the parameters of the resulting code, and compare our resulting non-linear codes with currently best known linear error-correcting codes.

Table of Contents

1	Introduction	1
2	Review of Basic Properties	4
2.1	Elliptic and Hyperelliptic Curves	4
2.2	Witt Vectors and the Greenberg Transform	8
3	Minimal Degree Liftings	14
3.1	Elliptic Curve Lifts	15
3.2	Hyperelliptic Curve Lifts	17
3.3	Algorithms	21
3.4	Results and Conjectures	30
3.4.1	Non-singular Affine Curves	32
3.4.2	Singular Affine Curves	46
4	Applications to Coding Theory	51
4.1	Linear Codes over Fields and Rings	51
4.2	Previous Applications to Non-Linear Codes	55
4.3	Modern Applications to Non-Linear Codes	57
4.4	Algorithms	60
4.5	Examples and Results	71
	Bibliography	89

List of Tables

3.1	Percentage of polynomials that could lift to the third coordinate via the algorithm given they lifted to the second coordinate via the algorithm by field.	35
3.2	Percentage of polynomials that could lift to the third coordinate via the algorithm given they lifted to the second coordinate via the algorithm by genus.	38
3.3	Percentage of polynomials that do not lift over $\mathbf{W}_3(\mathbb{k})$ with $d = 5$ given that the coefficient of $x_0^{2(p-1)}$ in $f(x_0)^{(p-1)/2}$ is zero.	41
3.4	Percentage of hyperelliptic curves with the coefficient on $x_0^{2(p-1)}$ of $f(x_0)^{(p-1)/2}$ equal to zero that are able to lift via Algorithm 2.	44
4.1	Non-linear codes over $\mathbb{F}_p$ with highest information rate and their minimum Hamming distance generated from lifted hyperelliptic curves over $\mathbf{W}_2(\mathbb{F}_p)$	76
4.2	Non-linear codes over $\mathbb{F}_p$ with highest information rate and their minimum Hamming distance generated from lifted hyperelliptic curves over $\mathbf{W}_3(\mathbb{F}_p)$	77
4.3	Non-linear codes over $\mathbb{F}_3$ with highest information rate and their minimum Hamming distance generated from lifted singular curves over $\mathbf{W}_\ell(\mathbb{F}_3)$	86

List of Figures

3.1	Bar plot of the ability of hyperelliptic curves given by monic polynomials $f(x_0)$ to lift to Witt vectors of length three via the algorithm given that they lift to Witt vectors of length two.	36
3.2	Bar plot of the percentage of hyperelliptic curves given by monic polynomials $f(x_0)$ that <i>can</i> lift to Witt vectors of length three via the algorithm given that they lift to Witt vectors of length two. . . .	39
3.3	Bar plot of the percentage of hyperelliptic curves given by a monic polynomial $f(x_0)$ of degree five or six that <i>cannot</i> lift to Witt vectors of length three via the algorithm given that the coefficient on $x_0^{2(p-1)}$ of $f(x_0)^{(p-1)/2}$ is zero.	45
4.1	Histogram of the information rate of non-linear codes of code rank two with degree $d = 5$ over $\mathbb{F}_7$ from the ring of Witt vectors of length three.	79
4.2	Histogram of the information rate of non-linear codes of code rank one with degree $d = 4$ over $\mathbb{F}_5$ from the ring of Witt vectors of length three.	80
4.3	Histogram of the information rate of non-linear codes of code rank two with degree $d = 5$ over $\mathbb{F}_5$ from the ring of Witt vectors of length three.	81
4.4	Scatter plots of information rate versus minimum Hamming distance for characteristic $p = 3$ grouped by polynomial degree d	82
4.5	Scatter plots of information rate versus minimum Hamming distance for characteristic $p = 5$ grouped by polynomial degree d	83

4.6	Scatter plots of information rate versus minimum Hamming distance for characteristic $p = 7$ grouped by polynomial degree d	84
-----	--	----

Chapter 1

Introduction

In the past few decades there has been increased research interest in the interplay of coding theory and algebraic geometry. Initially these areas of research were separate, as coding theory limited itself to codes involving vector spaces and fields. However, in 1982 Goppa used algebraic geometry to generalize Reed-Solomon codes, while in 1999 Walker in (26) introduced the idea of constructing algebraic geometric codes over rings instead of fields, and proved several foundational results relating to error-correcting codes. Walker in (25) also showed that the well-known Nordstrom-Robinson code could be represented as the non-linear image of an algebraic geometric code over $\mathbb{Z}/4\mathbb{Z}$. These results laid the framework and increased the interest in finding other non-linear shadows of well-known linear codes.

In (23), Voloch and Walker used the theory of canonical liftings of elliptic curves to construct non-linear error-correcting codes, based on Walker's previous work. These canonical liftings were proved to give, when restricted to Witt vectors of length two, the best possible bounds for the parameters for the codes obtained by the general construction. These bounds were based on degrees of certain coordinate functions associated to the canonical lifting. Later Finotti proved in (7) (for characteristic $p \geq 3$) and (8) (for characteristic $p = 2$) that for Witt vectors of length 3 or more, the canonical liftings did not give the best bounds for the parameters of the codes

anymore, which motivated the definition of *minimal degree liftings*. These minimal degree liftings would have coordinate functions with minimal degrees, and therefore yield the best bounds. Moreover, this definition and applications were also generalized to *hyperelliptic curves*.

In (8), Finotti used these minimal degree liftings to construct *binary* error-correcting codes, and showed that the construction indeed yielded good codes. Although not having as an immediate practical application, non-binary codes are also studied and are of theoretical interest. Thus, our main goal here will be to study these minimal degree liftings for characteristic $p \geq 3$ and their applications to non-binary codes.

It is worth noting that these minimal degree liftings are also of interest, independent of their applications to coding theory. For instance, in (7), Finotti shows that minimal degree liftings in characteristic 3 of curves of genus 2 are concrete examples of *Mochizuki liftings*, a result that is cited in Mochizuki's controversial proof of the ABC Conjecture.

Thus, after an introduction to the necessary background material, such as the necessary theory from algebraic geometry regarding elliptic and hyperelliptic curves, canonical liftings, and the ring of Witt vectors, we first turn our attention to properties of the minimal degree liftings. More precisely, we perform an exploratory analysis of their properties, look at minimal degree liftings of *singular* curves, and implement several key findings by Finotti. We develop algorithms to efficiently compute *best known absolute minimal degree lifts*, which are minimal degree liftings whose degrees meet the lower bounds, thus giving the best possible codes, for both hyperelliptic and singular curves. From this exploration we establish several conjectures and a few theoretical results regarding the ability to construct such lifts.

Once we establish the algebraic geometric results, we discuss the applications to coding theory. We start by expanding Voloch and Walker's and Finotti's applications to non-binary codes. We adapt their techniques to this new setting (characteristic greater than 2), in particular using generalizations of *Gray map* given in (12), (27),

and (24). Then, we implement the code for the construction of the resulting error-correcting codes and calculation of their parameters, examine the large amount of data obtained using these methods, and compare some of the results to the currently best known linear error-correcting codes.

Chapter 2

Review of Basic Properties

In this chapter we review some needed results regarding elliptic curves and hyperelliptic curves. We begin with a brief review of ordinary elliptic curves, followed by hyperelliptic curves. We also discuss some of their properties.

2.1 Elliptic and Hyperelliptic Curves

In this section we review basic properties of elliptic and hyperelliptic curves. Most properties described for elliptic curves can be found, with proofs, in (19). We will include definitions and results that will be relevant to later sections for the sake of completeness.

Let $E/\mathbb{k}$ be an elliptic curve over a perfect field $\mathbb{k}$ of characteristic $p > 0$. If $p \neq 2, 3$ then we may assume that the curve is given by the equation

$$E/\mathbb{k} : \quad y_0^2 = x_0^3 + a_0x_0 + b_0, \tag{2.1}$$

with $a_0, b_0 \in \mathbb{k}$. Equation (2.1) is referred to as the Weierstrass form of E . For $p = 2$ or $p = 3$ the Weierstrass forms are different, but we omit them here. They can be found in III.1 of (19).

By definition, E must be nonsingular, i.e., there are no points on E such that they satisfy equation (2.1) and the partial derivative equations $2y_0 = 0$ and $3x_0^2 + a_0 = 0$. This is equivalent to the saying that the discriminant of E , given by $\Delta = -16(4a_0^3 + 27b_0^2)$, is not equal to 0.

The $\mathbb{k}$ -rational points on E form a group, and we note that E is said to be *ordinary* if $E[p] = \mathbb{Z}/p\mathbb{Z}$, where $E[p]$ denotes the p -torsion subgroup (over the algebraic closure of $\mathbb{k}$). We also define the *Hasse invariant* to be the coefficient on x_0^{p-1} of the right-hand side of (2.1) raised to the power $(p-1)/2$. If E is ordinary, then the Hasse invariant is non-zero, otherwise it is zero. We also observe that elliptic curves have *genus* $g = 1$. (See (19).)

One can define curves with similar equations for genus $g > 1$, in which case we obtain what are referred to as hyperelliptic curves.

Definition 2.1.1. Let $\mathbb{k}$ be a perfect field of characteristic $p > 0$ and $\bar{\mathbb{k}}$ be the algebraic closure of $\mathbb{k}$. A *hyperelliptic curve* C of genus $g \geq 1$ over $\mathbb{k}$ is an equation of the form

$$C/\mathbb{k}: \quad y_0^2 + h(x_0)y_0 = f(x_0), \quad (2.2)$$

where $h(x_0) \in \mathbb{k}[x_0]$ is a polynomial of degree at most g , $f(x_0) \in \mathbb{k}[x_0]$ is a monic polynomial of degree d where $d = 2g + 1$ or $2g + 2$, and there are no solutions $(\alpha, \beta) \in \mathbb{k}^2$ satisfying (2.2) and the partial derivative equations $2y_0 + h(x_0) = 0$ and $h'(x_0)y - f'(x_0) = 0$.

Elliptic curves are a special case of definition 2.1.1 when $\deg(f(x_0)) = 3$. Note that for hyperelliptic curves, the $\mathbb{k}$ -rational points are no longer guaranteed to form a group. If the $\text{char}(\mathbb{k}) \neq 2$, then we have the following result.

Lemma 2.1.2. *Let $C/\mathbb{k}$ be a hyperelliptic curve as defined by equation (2.2). If $\text{char}(\mathbb{k}) \neq 2$, then we may apply a change of variables $x_0 \rightarrow x_0$, $y_0 \rightarrow (y_0 - h(x_0)/2)$*

to transform C into the form $y_0^2 = \tilde{f}(x_0)$ where $\deg \tilde{f} = \deg f$. Moreover, an equation of the form $y_0^2 = \tilde{f}(x_0)$ is a hyperelliptic curve if and only if $(\tilde{f}, \tilde{f}') = 1$.

Proof. See (16) Lemma 2. □

From this Lemma one can see that elliptic curves are a special case of definition 2.1.1 when $\deg(f(x_0)) = 3$, and equation 2.1 has $h(x_0) = 0$ as $\text{char}(\mathbb{k}) \neq 2$. From now on we will only consider hyperelliptic curves over fields with $\text{char}(\mathbb{k}) > 2$ unless otherwise stated.

The notion of singularity also applies to hyperelliptic curves. For any curve C of the form (2.2), not necessarily hyperelliptic, a point on C that satisfies the partial derivatives $2y_0 + h(x_0) = 0$ and $h'(x_0)y - f'(x_0) = 0$ is said to be singular. Curves that contain singular points are not said to be hyperelliptic. More generally, curves that contain no singular points are referred to as smooth. It can also be shown that if $h(x_0) = 0$, then C is singular if and only if the discriminant of $f(x_0)$ is zero, i.e., $\gcd(f(x_0), f'(x_0)) \neq 1$.

Note that while equation (2.2) is for an affine curve, we will think of C as a projective curve, i.e., a projective variety of dimension one. The same is true for elliptic curves as well. We note that for the projective curve, elliptic curves have one point at infinity, while hyperelliptic curves have at most two points at infinity.

For elliptic curves, there is only one point at infinity $\mathcal{O} \in \mathbb{P}^2$, namely $\mathcal{O} = [0 : 1 : 0]$. All other points on the curve we associate with their affine coordinates (α, β) which corresponds to the projective point $[\alpha : \beta : 1]$. For hyperelliptic curves, we can perform the same homogenization to have a bijection between projective and affine points, though we must take more care for hyperelliptic curves.

Note that by definition, elliptic curves are smooth, i.e., there are no points on the curve that satisfy all partial derivative equations. This is true of the projective and affine model. However, this is not true for hyperelliptic curves. One does want, by definition, for a hyperelliptic curve to be smooth in similar fashion to elliptic curves. However, while the affine model is smooth, the projective model formed by

homogenizing the affine model with one extra variable, results in a point at infinity that is singular. This can be avoided by gluing together two smooth affine models for the hyperelliptic curve, that results in two points at infinity if $\deg(f(x_0)) = 2g + 2$ (called *real hyperelliptic curves*) and one point at infinity if $\deg(f(x_0)) = 2g + 1$ (called *imaginary hyperelliptic curves*). This nuance will become important in Chapter 4. When there is single point at infinity, we will denote it by $\mathcal{O}$, and when there are more than one point at infinity, we will denote them $\mathcal{O}_1$ and $\mathcal{O}_2$.

We will need to know the order of x_0 and y_0 at these points. More generally, the order of a function at a point is the number of zeros a function has minus the number of poles. In the case of elliptic curves, it is well known that $\text{ord}_{\mathcal{O}}(x_0) = -2$, and $\text{ord}_{\mathcal{O}}(y_0) = -3$. In the case of hyperelliptic curves, it depends on the number of points at infinity. We assume here that we are referring to the smooth projective model for hyperelliptic curves. For imaginary hyperelliptic curves, one has that $\text{ord}_{\mathcal{O}}(x_0) = -2$, while $\text{ord}_{\mathcal{O}}(y_0) = -d$. In the case of real hyperelliptic curves, so there are two points at infinity, one has $\text{ord}_{\mathcal{O}_i}(x_0) = -1$ and $\text{ord}_{\mathcal{O}_i}(y_0) = -\frac{d}{2}$ for $i = 1, 2$. This will become relevant in Chapter 4.

We will also need to discuss divisors for these curves. Recall that the divisor group of a curve C is the free abelian group generated by the points of C and is denoted $\text{Div}(C)$. A divisor $D \in \text{Div}(C)$ is a formal sum

$$D = \sum_{P \in C} n_P(P),$$

where $n_P \in \mathbb{Z}$ and $n_P = 0$ for all but finitely many $P \in C$. The degree of D is then defined as the sum of the n_P for all $P \in C$. We note that one can show that the set of divisors of degree 0 form a subgroup; this subgroup is of much interest for hyperelliptic curves due to its applications for cryptography.

With the concept of the order of a function at a point and divisors, we can discuss the divisor of a smooth curve $f \in \bar{\mathbb{k}}(C)^*$, the function field of the curve. We associate

to f the divisor $\operatorname{div}(f)$ given by

$$\operatorname{div}(f) := \sum_{P \in C} \operatorname{ord}_P(f)(P),$$

where $\operatorname{ord}_P(f)$ is the order of vanishing of f at P . We also say that any divisor $D = \sum_{P \in C} n_P(P) \in \operatorname{Div}(C)$ is *positive*, i.e., $D \geq 0$, if $n_P \geq 0$ for all $P \in C$. If, for any two divisors $D_1, D_2 \in \operatorname{Div}(C)$ we have $D_1 - D_2 \geq 0$, then we say that $D_1 \geq D_2$.

For any divisor $D \in \operatorname{Div}(C)$, the set of functions

$$\mathcal{L}(D) := \{f \in \overline{\mathbb{K}}(C)^* : \operatorname{div}(f) \geq -D\} \cup \{0\},$$

forms a finite-dimensional $\overline{\mathbb{K}}$ -vector space whose dimension is denoted by $\ell(D)$ (see (19) II.5.2b). When D is clear from the context, we denote the vector space by $\mathcal{L}$.

2.2 Witt Vectors and the Greenberg Transform

In this section we define the ring of Witt vectors over a field. Many of the elementary properties of Witt vectors can be found in Chapter 8 of (14), which we state here. We begin by introducing the notion of a Witt polynomial, and from there derive the addition and multiplication operations necessary for Witt rings.

Definition 2.2.1. Let p be a prime, and for any $n \in \mathbb{Z}_{\geq 0}$ the n -th Witt polynomial is defined as

$$W_n(X_0, \dots, X_n) := X_0^{p^n} + pX_1^{p^{n-1}} + \dots + p^{n-1}X_{n-1}^p + p^n X_n. \quad (2.3)$$

One can then show that there exists polynomials $S_n, P_n \in \mathbb{Z}[X_0, \dots, X_n, Y_0, \dots, Y_n]$ such that

$$W_n(S_0, \dots, S_n) = W_n(X_0, \dots, X_n) + W_n(Y_0, \dots, Y_n) \quad (2.4)$$

and

$$W_n(P_0, \dots, P_n) = W_n(X_0, \dots, X_n) \cdot W_n(Y_0, \dots, Y_n). \quad (2.5)$$

With equations (2.4) and (2.5) we can define the ring of Witt vectors. Let $\mathbb{k}$ be a field of characteristic p . The *ring of Witt vectors of length n* $\mathbf{W}_n(\mathbb{k})$ is the set of n -tuples $(x_0, x_1, \dots, x_{n-1})$, $x_i \in \mathbb{k}$ (called *Witt vectors*) with the usual definition of equality. For our operations, let $\mathbf{s} = (s_0, s_1, \dots, s_n)$ and $\mathbf{t} = (t_0, t_1, \dots, t_n)$ be Witt vectors, and define addition to be

$$\mathbf{s} + \mathbf{t} := (S_0(s_0, t_0), S_1(s_0, s_1, t_0, t_1), \dots, S_n(s_0, \dots, s_n, t_0, \dots, t_n))$$

and multiplication to be

$$\mathbf{s} \cdot \mathbf{t} := (P_0(s_0, t_0), P_1(s_0, s_1, t_0, t_1), \dots, P_n(s_0, \dots, s_n, t_0, \dots, t_n)).$$

To simplify notation, we will write

$$S_n(\mathbf{s}, \mathbf{t}) := S_n(s_0, \dots, s_n, t_0, \dots, t_n)$$

and

$$P_n(\mathbf{s}, \mathbf{t}) := P_n(s_0, \dots, s_n, t_0, \dots, t_n).$$

Defining addition and multiplication on $W_n(\mathbb{k})$ in this way makes $W_n(\mathbb{k})$ into a commutative ring with 1. If we take n to infinity, we get the more general *ring of Witt vectors* over $\mathbb{k}$, denoted $\mathbf{W}(\mathbb{k})$.

We will be working with Witt vectors of length 3. Once can obtain the following formulas for computing the second and third coordinate of the vectors S_n, P_n (note

that $S_0 = X_0 + Y_0$ and $P_0 = X_0 Y_0$):

$$S_1 = X_1 + Y_1 + \frac{X_0^p + Y_0^p - (X_0 + Y_0)^p}{p}, \quad (2.6)$$

$$P_1 = X_1 Y_0^p + X_0^p Y_1, \quad (2.7)$$

$$S_2 = X_2 + Y_2 + \frac{1}{p} \left(X_1^p + Y_1^p - \left(X_1 + Y_1 + \frac{X_0^p + Y_0^p - (X_0 + Y_0)^p}{p} \right)^p \right) + \frac{X_0^{p^2} + Y_0^{p^2} - (X_0 + Y_0)^{p^2}}{p^2}, \quad (2.8)$$

$$P_2 = X_2 Y_0^{p^2} + X_1^p Y_1^p + X_0^{p^2} Y_2 + \frac{X_1^p Y_0^{p^2} + X_0^{p^2} Y_1^p - (X_1 Y_0^p + X_0^p Y_1)^p}{p}. \quad (2.9)$$

Since we are working over finite fields with characteristic p , we will let $\bar{S}_n, \bar{P}_n \in \mathbb{F}_p[X_0, \dots, X_n, Y_0, \dots, Y_n]$ be the reductions modulo p of S_n, P_n respectively. (Note that S_n and P_n have integer coefficients.)

We now give some basic facts about Witt vectors. Firstly, the additive identity is the vector of all zeros, which we denote by $\mathbf{0}$, and the multiplicative identity is the vector with 1 in the first position followed by 0's, which we denote by $\mathbf{1} = (1, 0, \dots)$. We also note that while $\mathbb{k}$ is of characteristic p , the characteristic of $\mathbf{W}(\mathbb{k})$ is 0. Since the characteristic of $\mathbf{W}(\mathbb{k})$ is not 0, then $p = \mathbf{1} + \mathbf{1} + \dots + \mathbf{1}$ (where we add p many times) is an element of $\mathbf{W}(\mathbb{k})$, namely $(0, 1, 0, \dots)$. One can also show that $\mathbf{W}(\mathbb{k})$ is a local ring, with maximal ideal (p) .

Additionally, a Witt vector $\mathbf{a} = (a_0, a_1, \dots) \in \mathbf{W}(\mathbb{k})$ is a unit if and only if $a_0 \in \mathbb{k}$ is a unit. Hence, $\mathbf{a}$ is a unit if and only if $a_0 \neq 0$. There is also a natural lift of the p -th power Frobenius σ of $\mathbb{k}$ defined as $\sigma(a_0, a_1, \dots) = (\sigma(a_0), \sigma(a_1), \dots)$. The Frobenius gives us a *trace map* $\text{tr} : \mathbf{W}(\mathbb{F}_q) \rightarrow \mathbf{W}(\mathbb{F}_p)$, where $q = p^r$ for some $r \in \mathbb{Z}_{\geq 1}$, where for $\mathbf{x} \in \mathbf{W}(\mathbb{F}_q)$

$$\text{tr}(\mathbf{x}) := \mathbf{x} + \sigma(\mathbf{x}) + \dots + \sigma^{r-1}(\mathbf{x}). \quad (2.10)$$

Finally, if $\mathbb{k} = \mathbb{F}_p$ for some prime p , then the ring of Witt vectors of length ℓ , $\mathbf{W}_\ell(\mathbb{F}_p)$, is isomorphic to the residue class ring $\mathbb{Z}/p^\ell\mathbb{Z}$. The isomorphism is as follows. Let $\mathbb{Z}_p$ denote the p -adic integers. Using Hensel's Lemma, one can show that $\mathbb{Z}_p^*$ contains a cyclic group of $p - 1$ roots of unity that have distinct reductions modulo p . Moreover, the reduction modulo p is an isomorphism between this group of roots of unity and $\mathbb{F}_p^*$. Letting $\omega : \mathbb{F}_p \rightarrow \mathbb{Z}_p^*$ be the inverse of this isomorphism and setting $\omega(0) = 0$, we obtain the *Teichmüller character* (see section 5.8 of (11) for proofs). Then, for any Witt vector $\mathbf{x} = (x_0, x_1, \dots, x_{\ell-1}) \in \mathbf{W}_\ell(\mathbb{F}_p)$, we define the isomorphism $\varphi : \mathbf{W}_\ell(\mathbb{F}_p) \rightarrow \mathbb{Z}/p^\ell\mathbb{Z}$ as

$$\varphi(x) = \sum_{i=0}^{\ell-1} \omega(x_i) p^i \pmod{p^\ell}.$$

Moreover, one can expand the isomorphism to $\mathbf{W}(\mathbb{F}_p)$, the ring of Witt vectors of infinite length over $\mathbb{F}_p$, to get an isomorphism $\Phi : \mathbf{W}(\mathbb{F}_p) \rightarrow \mathbb{Z}_p$.

Due to the recursive nature of Witt vectors, performing computations with Witt vectors can be unintuitive and computationally expensive. To help make computations of Witt vectors more manageable, we will often make use of the Greenberg transform.

Definition 2.2.2. Let $\mathbf{f}(\mathbf{x}, \mathbf{y}) \in \mathbf{W}(\mathbb{k})[\mathbf{x}, \mathbf{y}]$. If $\mathbf{x}_0 = (x_0, x_1, \dots)$, $\mathbf{y}_0 = (y_0, y_1, \dots) \in \mathbf{W}(\mathbb{k}[x_0, y_0, x_1, y_1, \dots])$, then $\mathbf{f}(\mathbf{x}_0, \mathbf{y}_0) = (f_0, f_1, \dots) \in \mathbf{W}(\mathbb{k}[x_0, y_0, x_1, y_1, \dots])$ for $f_n \in \mathbb{k}[x_0, \dots, x_n, y_0, \dots, y_n]$ is the *Greenberg transform* of $\mathbf{f}$ and is denoted $\mathcal{G}(\mathbf{f})$. Moreover, if

$$\mathbf{C}/\mathbf{W}(\mathbb{k}) : \quad \mathbf{f}(\mathbf{x}, \mathbf{y}) = 0,$$

we define the Greenberg transform $\mathcal{G}(\mathbf{C})$ of $\mathbf{C}$ to be the (infinite dimensional) variety over $\mathbb{k}$ defined by the zeros of the coordinates f_n of $\mathcal{G}(\mathbf{f})$.

Essentially, instead of thinking of polynomials over the ring of Witt vectors over a finite field, we can think of Witt vectors of polynomials over a finite field. From the definition we can see that there is a bijection between $\mathbf{C}(\mathbf{W}(\mathbb{k}))$ and $\mathcal{G}(\mathbf{C})(\mathbb{k})$ as $\mathbf{f}(\mathbf{a}, \mathbf{b}) = 0$ if and only if $f_n(a_0, \dots, a_n, b_0, \dots, b_n) = 0$ for all n . Moreover, we have that

$$\mathcal{G}(\mathbf{x} + \mathbf{y}) = (S_0, S_1, \dots) \quad \text{and} \quad \mathcal{G}(\mathbf{x} \cdot \mathbf{y}) = (P_0, P_1, \dots).$$

Using the following theorem of Finotti, one can compute the coordinates of the Greenberg transform recursively:

Theorem 2.2.3. *Let $\mathbf{f}(\mathbf{x}, \mathbf{y}) \in \mathbf{W}(\mathbb{k})[\mathbf{x}, \mathbf{y}]$ and suppose that $\mathcal{G}(\mathbf{f}) = (f_0, f_1, \dots)$. If*

$$W_n(\mathbf{f}_0, \mathbf{f}_1, \dots, \mathbf{f}_n) \equiv \mathbf{f}^{\sigma^n}(W_n(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_n), W_n(\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n)) \pmod{p^{n+1}} \quad (2.11)$$

for some $\mathbf{f}_i \in \mathbf{W}(\mathbb{k})[\mathbf{x}_0, \dots, \mathbf{x}_i, \mathbf{y}_0, \dots, \mathbf{y}_i]$, then $\mathbf{f}_i$ reduces to f_i modulo p .

Proof. This is Theorem 3.2 of (9). □

To make the notion of the Greenberg transform clear, we demonstrate the difference between an elliptic curve $E/\mathbb{k}$, a lift $\mathbf{E}/\mathbf{W}_\ell(\mathbb{k})$ that reduces to E modulo p , and its Greenberg transformation $\mathcal{G}(\mathbf{E})/\mathbb{k}$. Suppose that $E/\mathbb{k}$ is an elliptic curve given by equation (2.1), i.e.,

$$E/\mathbb{k} : \quad y_0^2 = x_0^3 + a_0 x_0 + b_0,$$

and $\mathbf{E}/\mathbf{W}_\ell(\mathbb{k})$ is a lift to the ring of Witt vectors of length ℓ , given by the equation

$$\mathbf{E}/\mathbf{W}_\ell(\mathbb{k}) : \quad \mathbf{y}^2 = \mathbf{x}^3 + \mathbf{a}\mathbf{x} + \mathbf{b}. \quad (2.12)$$

that reduces to E modulo p . We note that $\mathbf{y}$ and $\mathbf{x}$ are indeterminates of the polynomial ring $\mathbf{W}_\ell(\mathbb{k})[\mathbf{x}, \mathbf{y}]$ and $\mathbf{a}, \mathbf{b} \in \mathbf{W}_\ell(\mathbb{k})$. In comparison, the Greenberg

transform $\mathcal{G}(\mathbf{E})$ is given by the equation

$$\begin{aligned} \mathcal{G}(\mathbf{E})/\mathbb{k}: \quad (y_0, y_1, \dots, y_{\ell-1})^2 = (x_0, x_1, \dots, x_{\ell-1})^3 + (a_0, a_1, \dots, a_{\ell-1})(x_0, x_1, \dots, x_{\ell-1}) \\ + (b_0, b_1, \dots, b_{\ell-1}). \end{aligned} \quad (2.13)$$

In this case, x_i, y_i for $i = 0, \dots, (\ell - 1)$ are indeterminates of the polynomial ring $\mathbb{k}[x_0, \dots, x_{\ell-1}, y_0, \dots, y_{\ell-1}]$ and $a_i, b_i \in \mathbb{k}$ for $i = 0, \dots, (\ell - 1)$. Moreover, note that y_0, x_0, a_0 , and b_0 given in equation (2.13) are the same as that given in equation (2.1). This distinction between the lift of a curve over the ring of Witt vectors and the Greenberg transform of a curve will become more important in Section 3.3.

The benefit of this theorem is that it means the coordinates of the Greenberg transform can be computed recursively. When working with polynomial rings over the ring of Witt vectors, it can become computationally demanding computing S_i and P_i for $1 \leq i \leq n$ in order to compute the sum and product of elements of these polynomial rings. However, using Theorem 2.2.3, Finotti was able to compute explicit equations for the second and third coordinate of the Greenberg transform. We omit them here due to their length, but they can be found in (9).

Chapter 3

Minimal Degree Liftings

In this chapter we discuss the topic of lifts of curves. Let $\mathbb{k}$ be a field of characteristic $p > 0$. For any curve

$$C/\mathbb{k} : \quad y_0^2 = f(x_0), \tag{3.1}$$

we define the lift of C to be any curve

$$\mathbf{C}/\mathbf{W}(\mathbb{k}) : \quad \mathbf{y}^2 = \mathbf{f}(\mathbf{x}), \tag{3.2}$$

such that the reduction of $\mathbf{f}$ modulo p is f .

We first discuss results regarding the canonical lift of elliptic curves, followed by liftings of hyperelliptic curves. The first two sections provide the necessary background of the theory of these lifts, and the third section provides the algorithms implemented to generate these lifts. Finally we discuss results of an analysis of applying the lifting algorithm to monic polynomials of fixed degree over a finite field.

Many of the results of the first two sections are attributed to extensive research by Finotti (see (6), (7), (8)). Our algorithms are drawn from his research and guidance towards analyzing and understanding the strengths and limitations of this approach.

3.1 Elliptic Curve Lifts

Let $\mathbb{k}$ be a perfect field of characteristic $p > 0$. For an ordinary elliptic curve $E/\mathbb{k}$ as defined in Section 2.1, there exists a unique (up to isomorphism) elliptic curve $\mathbf{E}/\mathbf{W}(\mathbb{k})$ called the canonical lifting of E , and a map $\tau : E(\overline{\mathbb{k}}) \rightarrow \mathbf{E}(\mathbf{W}(\overline{\mathbb{k}}))$, a lift of points called the elliptic Teichmüller lift, such that

1. the reduction modulo p of $\mathbf{E}$ is E ;
2. if σ denotes the Frobenius of both $\mathbb{k}$ and $\mathbf{W}(\mathbb{k})$, then the canonical lifting of E^σ (elliptic curve obtained by applying σ to the coefficients of the equation that defines E) is $\mathbf{E}^\sigma$,
3. τ is an injective group homomorphism and a section of the reduction modulo p which we denote by π ,
4. if $\phi : E \rightarrow E^\sigma$ denotes the p -th power Frobenius, then there exists a map $\phi : \mathbf{E} \rightarrow \mathbf{E}^\sigma$ such that the diagram

$$\begin{array}{ccc} \mathbf{E}(\mathbf{W}(\mathbb{k})) & \overset{\Phi}{\dashrightarrow} & \mathbf{E}^\sigma(\mathbf{W}(\mathbb{k})) \\ \pi \downarrow \uparrow \tau & & \pi \downarrow \uparrow \tau^\sigma \\ E(\mathbb{k}) & \xrightarrow{\phi} & E^\sigma(\mathbb{k}) \end{array}$$

commutes.

This concept of canonical lifting of elliptic curves was first introduced by Deuring in (4) and then generalized to Abelian varieties by Serre and Tate in (15). This theory has been used in many interesting applications, such as counting rational points in ordinary elliptic curves, as in Satoh's (18), and counting torsion points of curves of genus $g = 2$, as in Poonen's (17) or Voloch's (22). We are concerned with its applications to coding theory, as we will discuss in further detail in Chapter 4.

We note that the above result guarantees the existence of the elliptic Teichmüller lift for ordinary elliptic curves. There has been considerable interest in understanding

the lift τ , along with its connection to the Greenberg transform. As seen in Section 2.2, for any curve $\mathbf{E}$ there is a Greenberg transform $\mathcal{G}(E)$, from which we are able to get a bijection of points $\gamma : \mathbf{E}(\mathbf{W}(\bar{\mathbb{k}})) \rightarrow \mathcal{G}(\mathbf{E})(\bar{\mathbb{k}})$. Of interest has been computing the degrees of the Witt coordinate functions of $\gamma \circ \tau = ((x_0, x_1, x_2, \dots), (y_0, y_1, y_2, \dots))$.

Voloch and Walker first studied these Witt coordinate functions, and found that if one begins with an ordinary elliptic curve $E/\mathbb{k}$ over a perfect field and constructs the canonical lift $\mathbf{E}/\mathbf{W}(\mathbb{k})$, then it must be that $\deg x_1 < 3p$ and $\deg y_1 < 4p$. Note here that degree refers to the degree of the field extension $\mathbb{k}(E)$ over $x_1^*(\mathbb{k}(E))$ and $y_1^*(\mathbb{k}(E))$, where x_1^* and y_1^* are the respective pullbacks. Conversely, if $\mathbf{E}$ is any elliptic curve defined over $\mathbf{W}_2(\mathbb{k})$ with reduction E and the projection given by reduction from $\mathcal{G}(\mathbf{E})$ to E admits a section τ in the category of $\mathbb{k}$ -schemes over $E \setminus \{\mathcal{O}\}$ given by $(x_0, y_0) \mapsto (\mathbf{x}, \mathbf{y}) = ((x_0, x_1), (y_0, y_1))$ where x_1, y_1 are regular away from $\mathcal{O}$ and satisfy $\deg x_1 < 3p$ and $\deg y_1 < 4p$, then $\mathbf{E}$ is the canonical lifting of E , τ is regular at $\mathcal{O}$, and is the elliptic Teichmüller lift of E (see Proposition 4.2 in (23)).

This provided not only an upper bound on the degrees of the coordinate functions, but also a formula that can be used to compute the coordinate functions; namely it gives that $x_1 \in \mathbb{k}[x_0]$ and (for $p \neq 2$)

$$\frac{dx_1}{dx_0} = A^{-1}y_0^{p-1} - x_0^{p-1}, \quad (3.3)$$

where A is the Hasse invariant of E , i.e., the coefficient of x_0^{p-1} in $f(x_0)^{(p-1)/2}$ where $f(x_0) = x_0^3 + a_0x_0 + b_0$. (If $p = 3$, then $f(x_0)$ has a slightly different formula, as mentioned in Section 2.1.)

In a similar fashion, Finotti expanded this idea to the third coordinate in (6), showing that if $\mathbf{E}/\mathbf{W}_3(\mathbb{k})$ is an elliptic curve with reduction E and there is a section τ over $E \setminus \{\mathcal{O}\}$ of the reduction from $\mathcal{G}(\mathbf{E})$ to E in the category of $\mathbb{k}$ -schemes that satisfies the conditions of Voloch and Walker, then $\mathbf{E}$ and τ are the canonical lift and elliptic Teichmüller lift, respectively, if and only if the degree in x_0 of $x_0^{p^2}x_2 - (3/4)x_1^{2p}$ is less than or equal to $(5p^2 - 1)/2$. This led to a way to recursively compute the

canonical lift to the third coordinate using the formula (for $p \neq 2$)

$$\frac{dx_2}{dx_0} = A^{-(p+1)}y_0^{p^2-1} - x_0^{p^2-1} - x_1^{p-1}\frac{dx_1}{dx_0}. \quad (3.4)$$

The benefit of these formulas is that it allowed the formation of an algorithm to compute the canonical lift $\mathbf{E}/\mathbf{W}_3(\mathbb{k})$ given an elliptic curve $E/\mathbb{k}$ such that the Witt coordinate polynomials of $\mathcal{G}(\mathbf{E})$ were minimized. However, Finotti found that if one simply integrates $\frac{dx_2}{dx_0}$ and does not ensure that the degree condition is met, then the algorithm still provides a section of the reduction π . This algorithm was given in Section 6 of (6), which we define as the *minimal degree lift* of an elliptic curve.

3.2 Hyperelliptic Curve Lifts

Work has been done by Finotti in generalizing the lifting algorithm to compute minimal degree lifts for hyperelliptic curves. In the case of hyperelliptic curves, there is no canonical lifting. But we still want to find liftings having small degrees, in part due to applications to error-correcting codes, as we shall discuss in Chapter 4. To that end, we introduce the following definitions and results from (7).

Let $C/\mathbb{k}$ be a hyperelliptic curve as defined by equation (3.1) and let also $\mathbf{C}/\mathbf{W}(\mathbb{k})$ be given by (3.2). We let U denote the affine part of C and $\mathbf{U}$ the affine part of $\mathbf{C}$.

Definition 3.2.1. Let $\mathbf{C}/\mathbf{W}(\mathbb{k})$ be as defined by equation (3.2) with reduction $C/\mathbb{k}$ modulo p . A *hyperelliptic lift* (of points) from C to $\mathbf{C}$ is a regular map $\nu : U(\overline{\mathbb{k}}) \rightarrow \mathbf{U}(\mathbf{W}(\overline{\mathbb{k}}))$ given by

$$\nu(x_0, y_0) = ((x_0, F_1, F_2, \dots), (y_0, G_1, G_2, \dots)), \quad (3.5)$$

such that $G_i = y_0 H_i$ and $F_i, H_i \in \mathbb{k}[x_0]$ for all i . We also let $F_0 := x_0$, $G_0 := y_0$, $H_0 := 1$.

The name hyperelliptic lift comes from the fact that these lifts commute with the hyperelliptic involutions $(x, y) \mapsto (x, -y)$.

We will be examining lifts where the coordinate functions have poles of small order at infinity. To simplify matters, we will refer to “degrees” instead of orders at poles, since we will be dealing with polynomials over affine points. These polynomials are in x_0 and y_0 , so for the next few results we will redefine degree so it is clear what we mean when minimizing the degree of a polynomial (for elliptic curves, minimizing the degree of the lift is equivalent to minimizing the degree of the functions x_1, x_2 in terms of x_0 in the typical sense). We define $\deg x_0 := 1$ and $\deg y_0 := d/2$, as $y_0^2 = f(x_0)$. One can think of these degrees as weights that we are trying to minimize. When considering polynomials in x_0 and y_0 , calculating the degrees for the polynomials is similar to counting the typical degree of a polynomial. For example, $\deg y_0^2 = d$.

Note that this definition of degrees for x_0 and y_0 are the same as *half* of their degrees as functions in the function field of the curves, as in Section II.2 of (19), i.e., $\deg(x_0) = [\mathbb{k}(C) : \mathbb{k}(x_0)]/2$ and $\deg(y_0) = [\mathbb{k}(C) : \mathbb{k}(y_0)]/2$, where $\mathbb{k}(C)$ denotes the function field of the curve.

Definition 3.2.2. Let C be an equation of the form (3.1) and $\mathbf{C}$ a lift of C . A *minimal degree lift from $C/\mathbb{k}$ to $\mathbf{C}/\mathbf{W}_2(\mathbb{k})$ with respect to $\mathbf{x}$* is a hyperelliptic lift of points

$$\nu(x_0, y_0) = ((x_0, F_1), (y_0, G_1)),$$

where $\deg F_1$ is minimal. Inductively, a minimal degree lift from C to $\mathbf{C}/\mathbf{W}_{n+1}(\mathbb{k})$ with respect to $\mathbf{x}$ is a hyperelliptic lift of points

$$\nu(x_0, y_0) = ((x_0, F_1, \dots, F_n), (y_0, G_1, \dots, G_n)),$$

whose reduction modulo p^n is a minimal degree lift from C to $\mathbf{C}/\mathbf{W}_n(\mathbb{k})$ and $\deg F_n$ is minimal.

Definition 3.2.3. Let C be a hyperelliptic curve as defined by 2.1.1. An *absolute minimal degree curve modulo p^2 over C (with respect to $\mathbf{x}$)* is a curve $\mathbf{C}/\mathbf{W}_2(\mathbb{k})$ which reduces to C modulo p , and satisfies the following property. Let

$$\nu(x_0, y_0) = ((x_0, F_1), (y_0, G_1))$$

be a minimal degree lift from C to $\mathbf{C}$, and let $\tilde{\mathbf{C}}/\mathbf{W}_2(\mathbb{k})$ be any curve that reduces to C modulo p . Then for any minimal degree lift

$$\tilde{\nu}(x_0, y_0) = ((x_0, \tilde{F}_1), (y_0, \tilde{G}_1)),$$

from C to $\tilde{\mathbf{C}}$, we have $\deg \tilde{F}_1 \geq \deg F_1$.

Inductively, an absolute minimal degree curve modulo p^{n+1} over C with respect to $\mathbf{x}$ is a curve $\mathbf{C}/\mathbf{W}_{n+1}(\mathbb{k})$ whose reduction modulo p^n is an absolute minimal degree curve modulo p^n over C , satisfying the following property. Let

$$\nu(x_0, y_0) = ((x_0, F_1, \dots, F_{n-1}, F_n), (y_0, G_1, \dots, G_{n-1}, G_n)),$$

be a minimal degree lift from C to $\mathbf{C}$, and let $\tilde{\mathbf{C}}/\mathbf{W}_{n+1}(\mathbb{k})$ be any curve whose reduction modulo p^n is equal to the reduction modulo p^n of $\mathbf{C}$. Then, for a minimal degree lift

$$\tilde{\nu}(x_0, y_0) = ((x_0, F_1, \dots, F_{n-1}, \tilde{F}_n), (y_0, G_1, \dots, G_{n-1}, \tilde{G}_n)),$$

from C to $\tilde{\mathbf{C}}$, we have $\deg \tilde{F}_n \geq \deg F_n$. In this case we call the minimal degree lift ν from C to $\mathbf{C}$ an absolute minimal degree lift of points (modulo p^{n+1}).

The following results of Finotti give upper and lower bounds to the absolute minimal degree lifts.

Proposition 3.2.4. *Let $C/\mathbb{k}$ and $\mathbf{C}/\mathbf{W}(\mathbb{k})$ be a hyperelliptic curve and its lift, respectively. Then there exists a unique minimal degree lift*

$$\nu = ((x_0, F_1, F_2, \dots), (y_0, y_0 H_1, y_0 H_2, \dots)),$$

from C to $\mathbf{C}$ with respect to $\mathbf{x}$, and we have

$$\deg F_n \leq \frac{dp^n + (d-2)}{2},$$

and

$$\deg H_n \leq \frac{(n(d-2) + d)p^n + n(d-2)p^{(n-1)} - d}{2},$$

for all $n \geq 0$.

Proof. This is Proposition 2.2 from (7). □

Theorem 3.2.5. *Let*

$$\nu = ((x_0, F_1, \dots, F_n), (y_0, G_1, \dots, G_n)),$$

where $n \geq 1$, be a lift of points (not necessarily hyperelliptic) from the affine part of C , given by equation (3.1), to the affine part of a lift $\mathbf{C}$, given by equation (3.2), where $F_i \in \mathbb{k}[x_0]$ with

$$\deg F_i = \frac{dp^i - (d-2)}{2},$$

for $i = 0, \dots, (n-1)$. Then, $\deg F_n \geq (dp^n - (d-2))/2$. If equality holds, then

$$\frac{dF_n}{dx_0} = A^{-(p^n-1)/(p-1)} f(x_0)^{(p^n-1)/2} - \sum_{i=0}^{n-1} F_i^{p^{n-i}-1} \frac{dF_i}{dx_0}, \quad (3.6)$$

where A is the (necessarily non-zero) coefficient of x_0^{p-1} in $f(x_0)^{(p-1)/2}$.

Proof. This is Theorem 2.4 in (8). □

We define A in Theorem 3.2.5 as the *Hasse invariant* of the hyperelliptic curve. Note that it matches the conventional definition for the Hasse invariant of an elliptic curve.

We note that Proposition 3.2.4 establishes that there is a unique minimal degree lift with an upper bound on the degree of F_n and H_n , while Theorem 3.2.5 establishes the best lower bound one can expect to get with an absolute minimal degree lift, namely

$$\deg F_n = \frac{dp^n - (d - 2)}{2} \quad (3.7)$$

for all $n \geq 1$. Moreover, it establishes basic conditions that are required for such a lifting to exist starting with $n = 1$. Firstly, since $\mathbb{k}$ is a field, we must have that the Hasse invariant, i.e, the coefficient of x_0^{p-1} in $f(x_0)^{(p-1)/2}$, is non-zero. Secondly, we must have that the coefficient of x_0^{ip-1} in $f(x_0)^{(p-1)/2}$ is equal to 0 for all $i \neq 1$, so that we may be able to compute the antiderivative of dF_1/dx_0 over characteristic p using equation (3.6).

3.3 Algorithms

In this section we describe the algorithms that were written to attempt construction of an absolute minimal degree lift for both elliptic and hyperelliptic curves. Due to applications to coding theory we are only interested in lifts that achieve the lower bound from Theorem 3.2.5. We will discuss these applications later on in Section 4.2. All algorithms were written in the computer algebra system (CAS) SageMath and can be found at

<https://github.com/justingroves/absolute-minimal-degree-lifts>.

We note that in principle, construction of absolute minimal degree lifts with the lower bound is not guaranteed. Finotti in (6) proved that such a lift does exist where the curve C has genus $g = 1$ (i.e., elliptic curves) up to Witt vectors of length three,

however for $g > 1$ it has not been proven. Thus our algorithms attempt to construct codes with the degree of F_i equal to the lower bound, but they may fail. We discuss this further in Section 3.4.

By Theorem 3.2.5, we can only attempt to construct an absolute minimal degree lift of hyperelliptic curves if the initial conditions described at the end of Section 3.2 are satisfied. Namely, we need that the Hasse invariant is non-zero, and that equation (3.6) for $n = 1$ is integrable over $\mathbb{k}$. If neither of these conditions are satisfied, we do not attempt to construct a lift satisfying the lower bound.

First we discuss the algorithm for elliptic curves. Finotti in (6) derived the algorithms to compute the minimal degree lift to the third coordinate for characteristic $p \neq 2, 3$. Finotti originally wrote these algorithms in Magma, another CAS, which can be found at the link

<https://github.com/lrfinotti/witt>.

Since Magma is not open source, we transcribed these functions to SageMath. Here we summarize those techniques and then expand upon them to compute the minimal degree lift in the hyperelliptic curve case.

The first thing to note is that SageMath is a CAS, so many algebraic objects (such as rings, fields, quotient spaces, etc.) are built into the language and are easily assessed and manipulated. However, the ring of Witt vectors has not been implemented as of version 10.0. This means that the class of Witt vectors are not implemented, and as such the algorithms needed to add and multiply Witt vectors together must be implemented by hand. This further complicates matters when one wants to work with a polynomial ring over the ring of Witt vectors.

To compensate for this, instead of working with Witt vectors directly, i.e., as an immutable vector, we instead work with a list object whose elements are from the base ring from which the ring of Witt vectors is derived. In the case of polynomial rings over Witt vectors, we work with the Greenberg transform to avoid the scenario where our indeterminates are Witt vectors. Since Witt vectors are not implemented

in SageMath, there is currently no way to build a polynomial ring object over the ring of Witt vectors.

The first file, `witt.sage`, includes functions to add, subtract, multiply, and invert Witt vectors of arbitrary finite length over a ring. It also includes functions to compute the Frobenius, trace, and raise a Witt vector to an arbitrary power. It is worth noting that while the ring of Witt vectors has not been implemented in SageMath, the p -adic integers $\mathbb{Z}_p$ have been. As such, occasionally it is computationally faster to first convert a Witt vector (if it is over $\mathbb{F}_p$) to an element of $\mathbb{Z}_p$ and then perform any desired computations in $\mathbb{Z}_p$ before converting back to an element of $\mathbf{W}_\ell(\mathbb{F}_p)$. Functions to convert back and forth between these two objects are also included in `witt.sage`.

The next file, `lift.sage`, includes functions to compute the coordinate polynomials of the composition of the Greenberg transform with the elliptic Teichmüller lift $\gamma \circ \tau$, given an elliptic curve $E/\mathbb{k}$ by equation (2.1), along with the coordinates of **a** and **b** after applying the Greenberg transform. Many of the functions in this file are auxiliary functions used to help efficiently compute the coordinate polynomials of the Greenberg transform $\mathcal{G}(E)$ and are a selection of the algorithms derived from the work in (9) and (10). We omit the details of these algorithms, but we include the details of the algorithm to compute the coordinate polynomials of $\gamma \circ \tau$. The function `lift` takes in $a_0, b_0 \in \mathbb{F}_q$ from equation 2.1, and n , the position of the Witt vectors to compute the lifts up to. This corresponds to Witt vectors of length $n + 1$, i.e., finds the canonical lift over $\mathbf{W}_{n+1}(\mathbb{k})$. Since we are concerned with lifts to the second and third coordinates and Witt vectors begin at position 0, this corresponds to $n = 1$ (i.e., $\mathbf{W}_2(\mathbb{k})$) and $n = 2$ (i.e., $\mathbf{W}_3(\mathbb{k})$).

The general idea of the algorithm is to take equation (2.13) and expand it on both the left and right hand side by performing the needed operations over the ring of Witt vectors (squaring on the left hand side, and adding and multiplying the vectors on the right hand side), leaving one vector on both the left and the right hand side. Then, use the formulas for dx_1/dx_0 and dx_2/dx_0 recursively to solve for the unknown

coordinates via a linear system. For example, when solving for the second coordinate ($n = 1$), we have the equation of $\mathbf{E}/\mathbf{W}_2(\mathbb{k})$, namely

$$(y_0, y_1)^2 = (x_0, x_1)^3 + (a_0, a_1)(x_0, x_1) + (b_0, b_1),$$

where $a_0, b_0 \in \mathbb{k}$ are known. Expanding gives

$$(y_0, 2y_0^p y_1) = \left(x_0^3 + a_0 x_0 + b_0, 3x_0^{2p} x_1 + a_0^p x_1 + a_1 x_0^p + b_1 + \frac{1}{p} (x_0^{3p} + a_0^p x_0^p + b_0^p - (x_0^3 + a_0 x_0 + b_0)^p) \right). \quad (3.8)$$

To solve for the unknowns, namely x_1, y_1, a_1 , and b_1 , integrate equation (3.3). Call this formal integral $\tilde{x}_1$. Since we perform this integration over a field of characteristic p , that means there are potential missing terms of the form x_0^{ip} (as their derivative is 0 (mod p)). From the results in (23), we know that the degree of the equation x_1 that gives the canonical lift (in x_0) is at most $(3p - 1)/2$. To account for potential missing terms in $\tilde{x}_1$, add missing powers of x_0^p to $\tilde{x}_1$ until reaching $x_0^{(3p-1)/2}$ with an indeterminate coefficient c_{ip} for $i = 0, 1$. Substitute $\tilde{x}_1$ in for x_1 in the right hand side of equation (3.8). In (23) it was shown that $y_1 = y_0 H_1(x_0)$ for some $H_1(x_0) \in \mathbb{k}[x_0]$. Taking the second coordinate from the left hand side of (3.8) gives $2y_0^p y_1 = 2y_0^p (y_0 H_1(x_0)) = 2y_0^{p+1} H_1(x_0) = 2(f(x_0))^{\frac{p+1}{2}} H_1(x_0)$. Dividing the right hand side by $2(f(x_0))^{\frac{p+1}{2}}$ (which contains no indeterminates) using polynomial long division will result in a remainder in x_0 with coefficients having indeterminates a_1, b_1, c_0, c_p . Since the canonical lift does exist, the remainder should be 0, i.e., the coefficients should all be 0. The coefficients then provide a linear system that can be solved for the indeterminates that make all the coefficients equal to 0. Once the indeterminates are known, they can be plugged back in to $\tilde{x}_1$ to give x_1 . This will also give $H_1(x_0)$, and thereby y_1 .

Once this process has been performed, we perform a similar operation following the results in (6) to determine a_2, b_2, x_2 , and y_2 . The pseudocode to perform this

algorithm in SageMath is given in Algorithm 1. We note that we assume that a_0 and b_0 give an ordinary elliptic curve, so the discriminant Δ and Hasse invariant are non-zero.

Algorithm 1 Minimal degree lift of elliptic curve ($p \neq 2, 3$)

```

1: procedure LIFT( $a_0, b_0, n$ )
2:    $p \leftarrow \text{char}(\mathbb{k})$ 
3:    $\text{res}_a \leftarrow [a_0]$ 
4:    $\text{res}_b \leftarrow [b_0]$ 
5:    $\text{res}_F \leftarrow [x_0]$   $\triangleright x_0$  is an indeterminate over  $\mathbb{k}$ 
6:    $\text{res}_H \leftarrow [1]$ 
7:    $f(x_0) \leftarrow x_0^3 + a_0x_0 + b_0$ 
8:    $\text{HI} \leftarrow \text{Hasse invariant}$   $\triangleright$  Coefficient of  $x_0^{p-1}$  in  $f(x_0)^{\frac{p-1}{2}}$ 
9:   for  $i \leftarrow 0, n-1$  do
10:     $M \leftarrow (3p^i - 1)/2$   $\triangleright$  Number of missing terms after integrating  $\frac{dF_i}{dx_0}$ 
11:     $P_i \leftarrow \mathbb{k}[x_0, y_0, a_{i+1}, b_{i+1}, c_0, \dots, c_M, H_{i+1}]$ 
12:    if  $i = 0$  then
13:       $g(x_0) \leftarrow f(x_0)^{\frac{p-1}{2}}$ 
14:    else
15:       $g(x_0) \leftarrow g(x_0)f(x_0)^{\frac{p-1}{2}}$ 
16:       $F_{i+1} \leftarrow (\text{HI})^{-\left(\frac{p^{i+1}-1}{p-1}\right)}g(x_0) - x_0^{p^{i+1}-1}$ 
17:      if  $i > 0$  then
18:         $F_{i+1} \leftarrow F_{i+1} - \sum_{j=1}^i \text{res}_F[j]^{p^{(i+1-j)}-1} \cdot \text{Derivative}(\text{res}_F[j])$ 
19:       $F_{i+1} \leftarrow \int F_{i+1} dx_0$ 
20:       $F_{i+1} \leftarrow F_{i+1} + \sum_{j=0}^M c_j x_0^{jp}$ 
21:       $v_a \leftarrow \text{res}_a$  append  $[a_{i+1}]$ 
22:       $v_b \leftarrow \text{res}_b$  append  $[b_{i+1}]$ 
23:       $v_F \leftarrow \text{res}_F$  append  $[F_{i+1}]$ 
24:       $v_G \leftarrow []$ 
25:      for  $j \leftarrow 0, i$  do
26:         $v_G[j] \leftarrow y_0 \cdot \text{res}_H[j]$ 
27:       $v_G[i+1] \leftarrow y_0 H_{i+1}$ 

```

Note that in our algorithm, we do not account for the needed additional terms to meet the degree requirement as stipulated in the end of Section 3.1 to ensure that τ is regular at $\mathcal{O}$. The reason is that we are concerned with lifts that minimize the degree of the coordinate polynomials, and omitting these terms still results in a lift of the affine points on the curve.

Algorithm 1 LIFT continued

```

28:   GTx  $\leftarrow$  GREENBERGTRANSFORM( $v_F^3 + v_a v_F + v_b$ )
29:   GTy  $\leftarrow$  GREENBERGTRANSFORM( $v_G^2$ )
30:   RHS  $\leftarrow$  GTx[ $i + 1$ ]
31:   LHS  $\leftarrow$  GTy[ $i + 1$ ]
32:   Remove coefficients from LHS having powers in  $H_{i+1}$ 
33:   Convert  $y_0^2$  in LHS to  $f(x_0)$ 
34:   Subtract portion of LHS in terms of  $x_0$  (not  $y_0$ ) from RHS
35:    $q(x_0) \leftarrow g(x_0)f(x_0)x_0$ 
36:   RHS  $\leftarrow \frac{1}{2}$ RHS
37:   Perform polynomial long division to get RHS =  $q(x_0)p(x_0) + r(x_0)$ 
38:    $v_{rem} \leftarrow []$ 
39:   for  $j \leftarrow 0, \deg(r(x_0))$  do
40:      $v_{rem}[j] \leftarrow$  Coefficient of  $x_0^j$  in  $r(x_0)$ 
41:   var  $\leftarrow [a_{i+1}, b_{i+1}, c_0, \dots, c_M]$ 
42:    $M \leftarrow$  Matrix where  $m_{i,j}$  is the coefficient of var[ $j$ ] from  $v_{rem}[j]$ 
43:    $t \leftarrow$  elements of  $v_{rem}$  evaluated at 0 for all indeterminates
44:    $s \leftarrow$  matrix solution to the equation  $sM = t$ 
45:   Append  $s[0]$  to  $\text{res}_a$   $\triangleright$  Newly found  $a_{i+1}$ 
46:   Append  $s[1]$  to  $\text{res}_b$   $\triangleright$  Newly found  $b_{i+1}$ 
47:   eval_vec  $\leftarrow [x_0, 0]$  append  $s$ 
48:   Append  $F_i(\text{eval\_vec})$  to  $\text{res}_F$ 
49:   Append  $p(\text{eval\_vec})$  to  $\text{res}_H$ 
return  $\text{res}_a, \text{res}_b, \text{res}_F, \text{res}_H$ 

```

Finotti also found that when solving for the missing coefficients of the x_0^{ip} terms, i.e., the c_i 's, there is a slight element of freedom. Since we are solving a matrix system, the solution we may find is not necessarily unique. Finotti found that by letting $c_{p^2} = 0$ in F_2 , the algorithm is still able to solve the system and produces and establishes a unique solution. This choice is not demonstrated in the pseudocode of Algorithm 1, but it can easily be implemented and is implemented in Section 3.4.

This same idea can be repeated in the instance of hyperelliptic curves. The formulas for $\frac{dF_n}{dx_0}$ and the bounds on the degrees of F_n allows one to repeat a similar algorithm. However, since for hyperelliptic curves we have that $f(x_0)$ is of arbitrary length, more care must be taken in computing the lift. In a similar fashion to elliptic curves, since these curves are over fields, we assume that the polynomial $f(x_0)$ is monic, and its lift is also monic. This is due to the fact that if $\deg(f(x_0)) = d$, then we are lifting to a curve of the form $\mathbf{y}^2 = \mathbf{f}(\mathbf{x})$ where $\mathbf{f}(\mathbf{x}) = \sum_{i=0}^d \mathbf{a}_i \mathbf{x}^i$. We have $\mathbf{a}_d$ is the leading coefficient of the lifted curve and $\mathbf{a}_d = (1, \dots)$, which is a unit in $\mathbf{W}(\mathbb{k})$, so we can multiply $\mathbf{f}(\mathbf{x})$ by its inverse and assume $\mathbf{f}(\mathbf{x})$ is monic. For all other coefficients, we denote them $\mathbf{a}_i = (a_{i,0}, a_{i,1}, \dots)$.

The pseudocode for the hyperelliptic curve case is given in Algorithm 2. We note that all we need to compute the lift is the polynomial $f(x_0)$. For elliptic curves, we assumed that our curve was ordinary. For hyperelliptic curves, our restrictions are different. The characteristic of the field is allowed to be 3. Secondly, $f(x_0)$ must have no repeated roots, so the discriminant $\Delta \neq 0$. Thirdly, we assume that $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ is integrable, so the coefficients of x_0^{ip-1} are 0 for all $i > 1$ (for $i = 1$ we need the Hasse invariant to be non-zero). If the above assumptions are not satisfied, then either the curve is singular (if $\Delta = 0$), or is not able to lift to an absolute minimal degree lift with lower bound described by Theorem 3.2.5. If the curve is singular, then it is still possible for the algorithm to produce a curve over $\mathbf{W}(\mathbb{k})$, however the theorems in (7) do not necessarily apply. More comments on lifts of singular curves are given in Section 3.4.

Algorithm 2 Minimal degree lift of hyperelliptic curves ($p \neq 2$)

```

1: procedure HYPER_LIFT( $f(x_0), n$ )
2:    $p \leftarrow \text{char}(\mathbb{k})$ 
3:    $d \leftarrow \deg(f(x_0))$ 
4:    $\text{res}_{\text{coef}}[i] \leftarrow []$ 
5:   for  $i \leftarrow 0, d-1$  do
6:      $\text{res}_{\text{coef}}[i] \leftarrow$  coefficients of  $x_0^i$  in  $f(x_0)$ 
7:    $\text{res}_F \leftarrow [x_0]$ 
8:    $\text{res}_H \leftarrow [1]$ 
9:    $\text{HI} \leftarrow$  Hasse invariant  $\triangleright$  Coefficient of  $x_0^{p-1}$  in  $f(x_0)^{\frac{p-1}{2}}$ 
10:  for  $i \leftarrow 0, n-1$  do
11:     $M \leftarrow \lfloor (dp^{i+1} - (d-2))/2p \rfloor$   $\triangleright$  Number of missing coefficients
12:     $P_i \leftarrow \mathbb{k}[x_0, y_0, a_{0,i+1}, a_{1,i+1}, \dots, a_{d-1,i+1}, c_0, \dots, c_M, H_{i+1}]$ 
13:    if  $i = 1$  then
14:       $g(x_0) = f(x_0)^{\frac{p+1}{2}}$ 
15:    else
16:       $g(x_0) = g(x_0)^p f(x_0)^{\frac{p-1}{2}}$ 
17:       $F_{i+1} \leftarrow (\text{HI})^{-\left(\frac{p^{i+1}-1}{p-1}\right)} g(x) - \sum_{j=0}^i F_j^{p^{i+1}-j-1} F'_j$ 
Ensure: Coefficients of  $x_0^{jp}$  in  $F_{i+1}$  are 0 for  $j > 1$ 
18:       $F_{i+1} \leftarrow \int F_{i+1} dx_0$ 
19:       $F_{i+1} \leftarrow F_{i+1} + \sum_{j=0}^M c_j x_0^{jp}$ 
20:       $v_a = []$ 
21:      for  $j \leftarrow 0, d-1$  do
22:         $v_a[j] \leftarrow \text{res}_{\text{coef}}[j]$  append  $a_{j,i+1}$  to  $v_a[j]$ 
23:       $v_F \leftarrow \text{res}_F$  append  $F_{i+1}$ 
24:       $v_G = []$ 
25:      for  $j \leftarrow 0, i$  do
26:         $v_G[j] \leftarrow y_0 \cdot \text{res}_H[j]$ 

```

Algorithm 2 HYPER_LIFT continued

```

27:   GTx  $\leftarrow$  GREENBERGTRANSFORM( $\sum_{j=0}^d v_a[j] \cdot v_G^j$ )
28:   GTy  $\leftarrow$  GREENBERGTRANSFORM( $v_G^2$ )
29:   RHS  $\leftarrow$  GTx[ $i + 1$ ]
30:   LHS  $\leftarrow$  GTy[ $i + 1$ ]
31:   Remove coefficients from LHS having powers in  $H_{i+1}$ 
32:   Convert  $y_0^2$  in LHS to  $f(x_0)$ 
33:   Subtract portion of LHS in terms of  $x_0$  (not  $y_0$  from RHS)
34:    $q(x_0) \leftarrow g(x_0)f(x_0)x_0$ 
35:   RHS  $\leftarrow \frac{1}{2}$ RHS
36:   Perform polynomial long division to get RHS =  $q(x_0)p(x_0) + r(x_0)$ 
37:    $v_{rem} \leftarrow []$ 
38:   for  $j \leftarrow 0, \deg(r(x_0))$  do
39:      $v_{rem}[j] \leftarrow$  coefficients of  $x_0^j$  in  $r(x_0)$ 
40:   var  $\leftarrow [a_{0,i+1}, a_{1,i+1}, \dots, a_{d-1,i+1}, c_0, \dots, c_M]$ 
41:    $M \leftarrow$  matrix where  $m_{i,j}$  is the coefficient of var[ $i$ ] from  $v_{rem}[j]$ 
42:    $t \leftarrow$  elements of  $v_{rem}$  evaluated at 0 for all indeterminates
43:    $s \leftarrow$  vector solution to the matrix equation  $sM = t$ 
44:   for  $j \leftarrow 0, d - 1$  do
45:     append  $s[j]$  to  $res_{coef}[j]$ 
46:   eval_vec  $\leftarrow [x_0, 0]$  append  $s$ 
47:   append  $F_i(\mathbf{eval\_vec})$  to  $res_F$ 
48:   append  $p(\mathbf{eval\_vec})$  to  $res_H$ 
49:    $v_1 \leftarrow [1]$ 
50:   for  $i \leftarrow 1, n$  do
51:      $v_1[i] \leftarrow 0$ 
52:   append  $v_1$  to  $res_{coef}$ 
return  $res_{coef}, res_F, res_H$ 

```

3.4 Results and Conjectures

In this section we go over empirical results of examining hyperelliptic curves of genus one through four over a small selection of finite fields. In this section, $\mathbb{k}$ will always refer to a finite field of characteristic at least $p = 3$. As noted before, in the case of elliptic curves, the canonical lifting guarantees that one is able to generate a lift satisfying the lower bound for $\deg(F_1)$. Moreover, as long as one does not care if the lift is regular at $\mathcal{O}$ and therefore is the elliptic Teichmüller lift, one can also obtain the lower bound for $\deg(F_2)$ (as proved by Finotti).

Proposition 3.2.4 establishes that given any lift $\mathbf{C}/\mathbf{W}(\mathbb{k})$ of a curve $C/\mathbb{k}$, one can find a minimal degree lift ν such that $\deg(F_n) \leq (dp^n + (d - 2))/2$. Theorem 3.2.5 establishes, though, that the best one can hope is that $\deg(F_n) = (dp^n - (d - 2))/2$ (where d is the degree of $f(x_0)$). This is a difference of $d - 2$, and it is not clear when the lower bound is achievable. We will refer to lifts over $\mathbf{W}_\ell(\mathbb{k})$ that have $\deg(F_i) = (dp^i - (d - 2))/2$ for $i = 1, \dots, \ell$ as *best known absolute minimal degree lifts*.

Recall that in constructing an absolute minimal degree lift for a hyperelliptic curve, the degree of F_i is minimized with each step. First, we minimize the degree of F_1 , and try to find a lift ν through solving a linear system. Then we repeat with F_2 . Our results found that indeed there are polynomials over finite fields that we are able to construct a best known absolute minimal degree lift over $\mathbf{W}_2(\mathbb{k})$, but are not able to do so over $\mathbf{W}_3(\mathbb{k})$.

Before we discuss these results, we first address a few preliminary observations regarding what polynomials we can hope to achieve best known absolute minimal degree lifts.

Proposition 3.4.1. *Let $\mathbb{k}$ be a field of characteristic p . If $C/\mathbb{k}$ is a hyperelliptic curve defined by equation (3.1) and $\deg(f(x_0)) \equiv 2 \pmod{p}$, then there does not exist a best known absolute minimal degree lift over $\mathbf{W}_\ell(\mathbb{k})$ for $\ell \geq 2$.*

Proof. Let $C/\mathbb{k} : y_0^2 = f(x_0)$ with lift $\mathbf{C}/\mathbf{W}_\ell(\mathbb{k}) : \mathbf{y}^2 = \mathbf{f}(\mathbf{x})$ and assume there exists a best known absolute minimal degree lift $\nu = ((x_0, F_1), (y_0, G_1))$ of C to $\mathbf{C}$. Let $\deg(f(x_0)) = d \equiv 2 \pmod{p}$. From Theorem 3.2.5, we have that

$$\frac{dF_1}{dx_0} = A^{-1}f(x_0)^{\frac{p-1}{2}} - x_0^{p-1}.$$

If the Hasse invariant of $f(x_0)$ is 0, then we are unable to compute A^{-1} , so we cannot obtain the best known absolute minimal degree lift. Assume then that $A \neq 0$. Then we see that

$$\frac{dF_1}{dx_0} = A^{-1}x_0^{d(\frac{p-1}{2})} + \dots,$$

where the omitted terms have degree less than $d(p-1)/2$. Note that $d(p-1)/2 \equiv p-1 \pmod{p}$, so $d(p-1)/2 = kp + p - 1 = p(k+1) - 1$ for some $k \in \mathbb{Z}_{\geq 0}$. This means that dF_1/dx_0 is not integrable over $\mathbb{k}$, as we are unable to integrate and divide the leading term by $p(k+1)$. This contradicts our assumption on ν , thus there is no hyperelliptic lift of C that is the best known absolute minimal degree lift. $\square$

This proposition shows that there are polynomials that we are unable to achieve the lower bound for any F_i for any hyperelliptic lift. Moreover, there are further restrictions imposed by the necessary requirement of the Hasse invariant to be non-zero. For example, consider the hyperelliptic curve

$$C/\mathbb{F}_5 : y_0^2 = x_0^5 + x_0.$$

The Hasse invariant of $f(x_0) = x_0^5 + x_0$ is equal to 0, so we are unable to use the algorithm to construct a lift of the curve, even though the degree of the polynomial is not congruent to 2 $\pmod{5}$.

With these concerns in mind, we set out to analyze curves of small genus over finite fields of small characteristic.

3.4.1 Non-singular Affine Curves

We begin with a discussion on non-singular affine curves, i.e., hyperelliptic curves. Let $y_0^2 = f(x_0)$, where $f(x_0) = \sum_{i=0}^d a_i x_0^i$. The two most time consuming computations in Algorithm 2 are the Greenberg Transform on lines 27 and 28, and solving the matrix system on line 43. As the characteristic of $\mathbb{k}$ and the degree of $f(x_0)$ increases, this significantly increases the computational time required to complete these steps. However, for relatively small prime p and degree d , computation time is quite quick and we were able to examine a large number of polynomials.

Since we were over a field, we only considered monic polynomials of fixed degree. Moreover, if $p \nmid d$, then we can examine the polynomial in depressed form and remove the $(d-1)$ -st term of the polynomial through the change of variable $x_0 \mapsto x_0 + \frac{a_{d-1}}{d}$. We attempted to compute the lift to the third coordinate, i.e., over $\mathbf{W}_3(\mathbb{k})$.

A preliminary analysis showed that if the Hasse invariant A is non-zero and $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ is integrable over $\mathbb{k}$ (i.e., the coefficient of x_0^{ip-1} is 0 for all $i \geq 1$), then a best known absolute minimal degree lift does exist over $\mathbf{W}_2(\mathbb{k})$. Thus while it has not been proven, we do have the following conjecture.

Conjecture 3.4.2. *Let $C/\mathbb{k}$ be a hyperelliptic curve with $\deg(f(x_0)) = d$ as in equation (3.1) where $\text{char}(\mathbb{k}) > 2$. If the Hasse invariant A is non-zero and $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ is integrable over $\mathbb{k}$, then there exists a hyperelliptic lift $\nu = ((x_0, F_1), (y_0, G_1))$ to $\mathbf{C}/\mathbf{W}_2(\mathbb{k})$ such that*

$$\deg F_1 = \frac{d(p-1) + 2}{2}.$$

Since the algorithm computes the coordinate polynomials recursively, this means that as long as the Hasse invariant and derivative conditions for $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ are satisfied, then a hyperelliptic lift over $\mathbf{W}_3(\mathbb{k})$ that achieves the best known minimal degree is possible. However, we found that there are polynomials that are able to get

the lower bound for the second coordinate using the algorithm, but are not for the third coordinate.

Note that for elliptic curves this does not happen! Finotti showed in (8) that if the Hasse invariant is non-zero (in which case we automatically have that $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ is integrable, since $\deg(f) = 3$), then we can get the best known minimal degrees for both the first and second coordinates.

Once a curve is able to obtain the lower bound for the second coordinate, there are two conditions in order to get the lower bound for the third coordinate. Firstly, it must be that

$$\frac{dF_2}{dx_0} = A^{-(p+1)}f(x_0)^{\frac{p^2-1}{2}} - \left(x_0^{p^2-1} + F_1^{p-1} \frac{dF_1}{dx_0} \right) \quad (3.9)$$

is integrable over $\mathbb{k}$. Secondly, we must be able to solve the linear system on the indeterminates to make the remainder of the polynomial long division zero. We found that in all our computations that while the first condition was *always* satisfied (see the conjecture below), there are instances for which the second condition was not.

Conjecture 3.4.3. *Let $\mathbb{k}$ is a field of characteristic $p > 2$ and $f \in \mathbb{k}[x_0]$ be a polynomial of degree at least 3, such that the coefficient A of x_0^{p-1} in $f(x_0)^{(p-1)/2}$ is non-zero, and there exists some $F_1 \in \mathbb{k}[x_0]$ such that $dF_1/dx_0 = A^{-1}f(x_0)^{(p-1)/2} - x_0^{(p-1)}$. Then, there is some $F_2 \in \mathbb{k}[x_0]$ such that $dF_2/dx_0 = A^{-p-1}f(x_0)^{(p^2-1)/2} - x_0^{p^2-1} - F_1^{p-1}dF_1/dx_0$.*

The conjecture is know for the case of $\deg(f) = 3$, since as we can take F_2 coming from the elliptic Teichmüller lift and was numerically verified for a very large number of cases.

As an example when the second condition is not satisfied, consider the hyperelliptic curve

$$C/\mathbb{F}_7 : \quad y_0^2 = x_0^7 + 4x_0^5 + 6x_0^4 + x_0^3 + 2.$$

It has absolute minimal degree lift over $\mathbf{W}_2(\mathbb{F}_7)$

$$\begin{aligned}(y_0, G_1)^2 = & (1, 0)(x_0, F_1)^7 + (0, 2)(x_0, F_1)^6 + (4, 5)(x_0, F_1)^5 + (6, 0)(x_0, F_1)^4 \\ & + (1, 3)(x_0, F_1)^3 + (0, 2)(x_0, F_1)^2 + (0, 4)(x_0, F_1) + (2, 4),\end{aligned}$$

where

$$\begin{aligned}F_1 = & 6x_0^{22} + 5x_0^{20} + 2x_0^{19} + 3x_0^{18} + x_0^{17} + 6x_0^{15} + 2x_0^{13} + 5x_0^{12} \\ & + 4x_0^{10} + x_0^9 + 6x_0^6 + x_0^5 + 4x_0^4 + 6x_0, \\ G_1 = & y_0(6x_0^{22} + 5x_0^{20} + 2x_0^{19} + 3x_0^{18} + x_0^{17} + 6x_0^{15} + 2x_0^{13} + 5x_0^{12} \\ & + 4x_0^{10} + x_0^9 + 6x_0^6 + x_0^5 + 4x_0^4 + 6x_0).\end{aligned}$$

However, when solving for the third coordinate, SageMath found that there was no matrix solution to force the remainder $r(x_0)$ of the polynomial long division to be zero. Hence being able to find the best known absolute minimal degree lift to the second coordinate does not guarantee that such a lift also exists to the third coordinate of $\mathbf{W}(\mathbb{k})$.

The question then arises of when can one expect to find a hyperelliptic lift to the third coordinate of $\mathbf{W}(\mathbb{k})$ with best known minimum degree of F_2 , given that one has a lift with best known minimum degree of F_1 . To answer this question, we examined all monic polynomials (in depressed form if possible) over the fields and degrees listed in Table 3.1. We ignored polynomials that did not meet the conditions to lift with best known absolute minimum degree of F_1 , i.e., polynomials where the Hasse invariant was zero or $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ was not integrable over $\mathbb{k}$. Our initial statistics can be found in Table 3.1, along with a visualization in Figure 3.1.

For most characteristics, if a curve was able to lift to the second coordinate with a best known minimum degree lift, then most likely that curve was able to lift similarly to the third coordinate via the algorithm. However, for characteristic three most polynomials were only able to lift via the algorithm to the second coordinate, and not

Table 3.1: Percentage of polynomials that could lift to the third coordinate via the algorithm given they lifted to the second coordinate via the algorithm by field.

Field	$\deg(f(x_0))$	% with no lift to $\mathbf{W}_3(\mathbb{F}_p)$	% with lift to $\mathbf{W}_3(\mathbb{F}_p)$
$\mathbb{F}_3$	6, 7, 9, 10	64.81%	35.18%
$\mathbb{F}_5$	5, 6, 8	12.79%	87.21%
$\mathbb{F}_7$	5, 6, 7	12.15%	87.84%
$\mathbb{F}_{11}$	5, 6	5.18%	94.82%
$\mathbb{F}_{13}$	5, 6	5.73%	94.27%
$\mathbb{F}_{17}$	5	2.84%	97.16%
$\mathbb{F}_{19}$	5	3.67%	96.32%

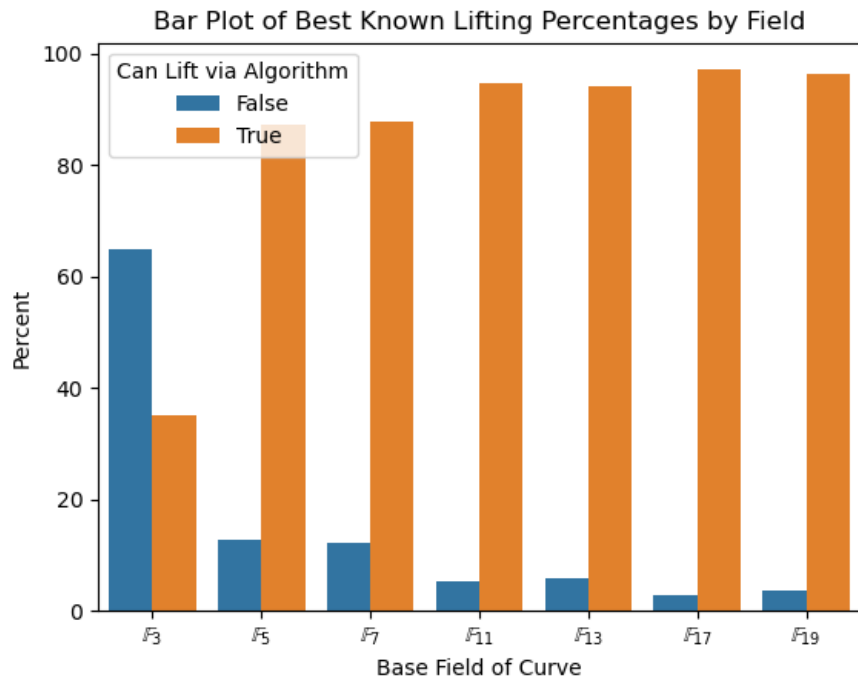


Figure 3.1: Bar plot of the ability of hyperelliptic curves given by monic polynomials $f(x_0)$ to lift to Witt vectors of length three via the algorithm given that they lift to Witt vectors of length two.

the third. As the characteristic of the field increased, though, the number of monic polynomials increased, and the percentage of polynomials that were not able to lift decreased. Note that for the fields $\mathbb{F}_{11}$, $\mathbb{F}_{13}$, $\mathbb{F}_{17}$, and $\mathbb{F}_{19}$ we only examined curves of genus 2. For $\mathbb{F}_3$, $\mathbb{F}_5$, and $\mathbb{F}_7$ we can break down these percentages by genus in Table 3.2.

It is interesting to note that for all characteristics examined save $p = 3$ we found that the majority of curves that have a best known lift to the second coordinate still have a best known lift to the third coordinate. However, when $p = 3$ and the genus of the curve was greater than two, that percentage flipped, with the majority of curves with a best known lift to the second coordinate were not able to lift. The data in Tables 3.1 and 3.2 came from curves over $\mathbb{F}_3$. We separately examined curves of genus two and three over $\mathbb{F}_9$ and we found that the same pattern held. After examining all possible monic polynomials, we found that for genus two 90.12% of the curves we were able to lift using the algorithm to Witt vectors of length three, while for genus three only 11.08% of polynomials were able to lift via the algorithm.

We then examined individual polynomials $f(x_0)$ over specific degrees to identify any pattern indicating if a curve could or could not be lifted to the third coordinate via the algorithm (meaning that the lower bound could not be achieved). Many of the techniques we utilized did not shed any insight on what was causing the matrix to not be solvable for the third coordinate. We examined the discriminant of the polynomial, and there was no pattern between those that did lift and those that did not. We also examined the roots of $f(x_0)$, but there was no pattern there either.

Finally, we tried interpolating on the coefficients of $f(x_0)$ to find if there was a polynomial condition for obtaining the lower bound (for the third coordinate). We set up a linear system of combinations of the coefficients up to degree five, and performed row reduction to examine if there was any combination of coefficients that was equal to a fixed number, indicating if a curve could lift to the best known minimal degree lift. (This was done since the conditions for obtaining the best known degree for the second coordinate can be expressed this way.) For almost all fields and degrees,

Table 3.2: Percentage of polynomials that could lift to the third coordinate via the algorithm given they lifted to the second coordinate via the algorithm by genus.

Field	Genus	% with no lift to $\mathbf{W}_3(\mathbb{F}_p)$	% with lift to $\mathbf{W}_3(\mathbb{F}_p)$
$\mathbb{F}_3$	2	22.22%	77.78%
$\mathbb{F}_3$	3	70.37%	29.63%
$\mathbb{F}_3$	4	66.87%	33.13%
$\mathbb{F}_5$	2	13.00%	87.00%
$\mathbb{F}_5$	3	12.71%	87.29%
$\mathbb{F}_7$	2	10.45%	89.55%
$\mathbb{F}_7$	3	12.43%	87.57%

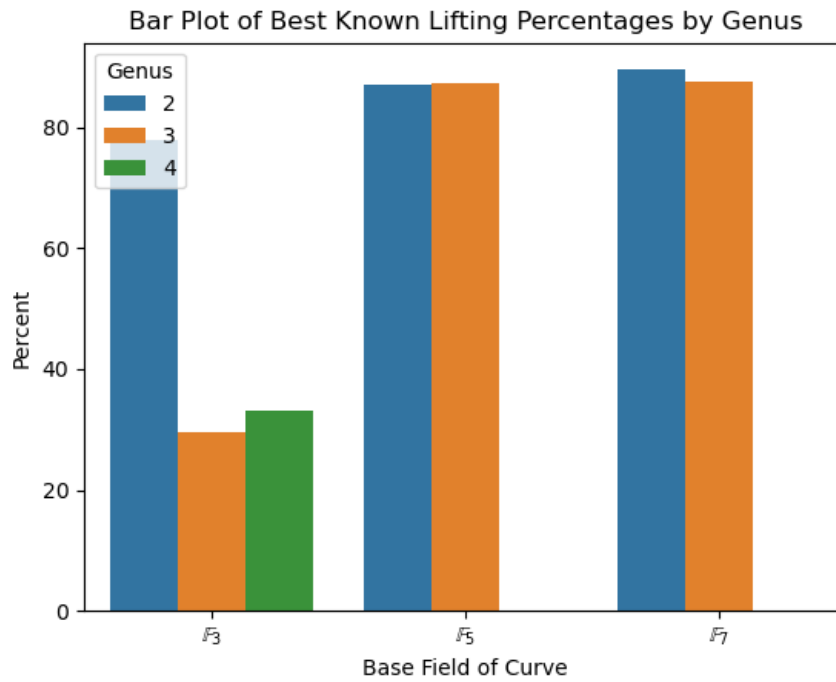


Figure 3.2: Bar plot of the percentage of hyperelliptic curves given by monic polynomials $f(x_0)$ that *can* lift to Witt vectors of length three via the algorithm given that they lift to Witt vectors of length two.

we could find no such condition. However, for $\mathbb{F}_7$ and $\mathbb{F}_{13}$ when $d = 5$, all monic depressed polynomials $f(x_0) \in \mathbb{k}[x_0]$ that were not able to lift via the algorithm were such that the coefficient of $x_0^{2(p-1)}$ in $f(x_0)^{(p-1)/2}$ was zero. It is interesting to note the similarity between these coefficient and that of the Hasse invariant (which we require to be non-zero) which is the coefficient of $x_0^{(p-1)}$ in $f(x_0)^{(p-1)/2}$.

Upon examining this condition, we found while the coefficient being zero was not enough to guarantee the polynomial was not able to lift via the algorithm, it was highly indicative for $d = 5$ that the polynomial was most likely not able to do so. In Table 3.3 we calculated the percentage of polynomials that were not able to lift via the algorithm, given that the coefficient of $x_0^{2(p-1)}$ on $f(x_0)^{(p-1)/2}$ was zero. Note that for $p = 3$, we have that $5 \equiv 2 \pmod{3}$, so there are no polynomials that were able to lift via the algorithm.

It is surprising that while such a low percentage of monic polynomials that satisfied the Hasse invariant and derivative conditions were not able to lift minimally, examining a single coefficient of $f(x_0)^{(p-1)/2}$ was highly indicative of whether the polynomial can be lifted to $\mathbf{W}_3(\mathbb{k})$ via the algorithm. Moreover, if this coefficient is non-zero, then the polynomial is able to lift minimally. Thus we have the following conjecture.

Conjecture 3.4.4. *Let $\mathbb{k}$ be a field of characteristic $p \geq 5$, and $C/\mathbb{k}$ be a hyperelliptic curve as in equation (3.1). If $\deg(f(x_0)) = 5$ and the coefficient on $x_0^{2(p-1)}$ of $f(x_0)^{(p-1)/2}$ is non-zero, then there exists a hyperelliptic lift $\nu = ((x_0, F_1, F_2), (y_0, G_1, G_2))$ to $\mathbf{C}/\mathbf{W}_3(\mathbb{k})$ such that for $i = 2, 3$*

$$\deg F_i = \frac{dp^i - (d - 2)}{2}.$$

This conjecture could be proved computationally for a fixed p : one simply has to compute the lift symbolically (in terms of the coefficients of f) and verify that only the Hasse invariant, the discriminant, and the coefficient of $x_0^{2(p-1)}$ in $f(x_0)^{(p-1)/2}$

Table 3.3: Percentage of polynomials that do not lift over $\mathbf{W}_3(\mathbb{k})$ with $d = 5$ given that the coefficient of $x_0^{2(p-1)}$ in $f(x_0)^{(p-1)/2}$ is zero.

Field	% of $f(x_0)$ that did not lift to $\mathbf{W}_3(\mathbb{k})$
$\mathbb{F}_5$	80.00%
$\mathbb{F}_7$	100.00%
$\mathbb{F}_{11}$	85.71%
$\mathbb{F}_{13}$	100.00%
$\mathbb{F}_{17}$	80.00%
$\mathbb{F}_{19}$	86.67%

appear in the denominators. But even with $p = 5$ this computation ran for weeks without finishing, so it still remains a conjecture.

The converse, unfortunately, is not true in general, as seen from the percentages in Table 3.3. However, in the cases of characteristics 5 and 7 we were able to find additional conditions allowing us to know exactly when a hyperelliptic curve with $d = 5$ is able to lift minimally or not. These results came from implementing Algorithm 2 with indeterminate values for the coefficients a_i for $f(x_0)$ and examining the corresponding linear system. We were able to prove the following results.

Theorem 3.4.5. *Let $\mathbb{k}$ be a field of characteristic 5 and $C/\mathbb{k}$ be a hyperelliptic curve as in equation (3.1), where $f(x_0) = \sum_{i=0}^5 a_i x_0^i$ is monic. Suppose that the coefficient on x_0^8 of $f(x_0)^2$ is zero. Then there exists a hyperelliptic lift $\nu = ((x_0, F_1, F_2), (y_0, G_1, G_2))$ to $\mathbf{C}/\mathbf{W}_3(\mathbb{k})$ with best known minimal degree lift if and only if $a_0 a_2 + a_1^2 = 0$.*

Theorem 3.4.6. *Let $\mathbb{k}$ be a field of characteristic 7 and $C/\mathbb{k}$ be a hyperelliptic curve as in equation (3.1), where $f(x_0) = \sum_{i=0}^5 a_i x_0^i$ is monic. If the coefficient on x_0^{12} of $f(x_0)^3$ is 0, then there does not exist a best known minimal degree lift $\nu = ((x_0, F_1, F_2), (y_0, G_1, G_2))$ to $\mathbf{C}/\mathbf{W}_3(\mathbb{k})$.*

Both of these propositions were *proved* computationally using SageMath. The idea was put the augmented matrix of the resulting linear system in row echelon form, and look at rows where we have all zeros (corresponding to the coefficients of the equation) followed by a single non-zero expression on the coefficients of f (corresponding to the free term of the equation). For a solution to exist, these non-zero expressions must be zero, and these gave us the conditions above.

Note that for characteristic 7 there was no additional condition except for the coefficient being zero.

These results seem to indicate that there could be conditions on the coefficients of $f(x_0)$ for all hyperelliptic curves to indicate when there does exist a best known minimal degree lift. Unfortunately, analyzing the augmented matrix system for each

characteristic and degree pair is inefficient, and for large characteristic and large degree infeasible.

Once we begin to look at hyperelliptic curves where $\deg(f(x_0)) > 5$, this pattern unfortunately disappears. When examining hyperelliptic curves with $d = 6$ the coefficient condition we have when $d = 5$ is still a strong indicator of when a polynomial is not able to lift minimally, but not nearly as strong as when $d = 5$ (see Table 3.4 and Figure 3.3).

When moving up to $d = 7$, we see $3(p - 1)$ start to appear as a power of x_0 in $f(x_0)^{(p-1)/2}$. At this point a hypothesis similar to even Conjecture 3.4.2 no longer holds when considering the coefficients of $x_0^{i(p-1)}$ for $i = 1, 2, 3$. For example, consider the hyperelliptic curve

$$C/\mathbb{F}_7 : \quad y_0^2 = x_0^7 + 5x_0^5 + 2x_0^4 + 5x_0^3 + 4x_0 + 3.$$

Computing $f(x_0)^3$ shows that the coefficients of x_0^6 , x_0^{12} , and x_0^{18} are non-zero, and there exists a best known absolute minimal degree lift via the algorithm to a curve over $\mathbf{W}_2(\mathbb{F}_7)$. However, the algorithm fails when trying to compute a lift over $\mathbf{W}_3(\mathbb{F}_7)$.

These results show that there is more work to be done in understanding why a best known absolute minimal degree hyperelliptic lift to the third coordinate is not possible even if there is one to the second coordinate. Moreover, there remains a question of if a best known minimal degree lift exists to the third coordinate, does that guarantee a lift to the fourth coordinate. It would be ideal if there is one unifying result to indicate when a hyperelliptic curve can be lifted minimally to any coordinate or not, but at this time it is not clear what that unifying result is, nor how to achieve it.

It is worth noticing the when performing some symbolic computations with curves of genus greater than one (i.e., leaving the coefficients of $f(x_0)$ as indeterminates), sometimes the discriminant of the curve appeared in the denominators of the formulas. This seems to never happen, with the same algorithm, for genus 1. In fact, this

Table 3.4: Percentage of hyperelliptic curves with the coefficient on $x_0^{2(p-1)}$ of $f(x_0)^{(p-1)/2}$ equal to zero that are able to lift via Algorithm 2.

Field	$\deg(f(x_0))$	% with no lift to $\mathbf{W}_3(\mathbb{F}_p)$	% with lift to $\mathbf{W}_3(\mathbb{F}_p)$
$\mathbb{F}_5$	5	80.00%	20.00%
	6	62.50%	37.50%
$\mathbb{F}_7$	5	100.00%	0.00%
	6	81.08%	18.92%
$\mathbb{F}_{11}$	5	85.71%	14.29%
	6	72.34%	27.66%
$\mathbb{F}_{13}$	5	100.00%	0.00%
	6	87.86%	12.14%

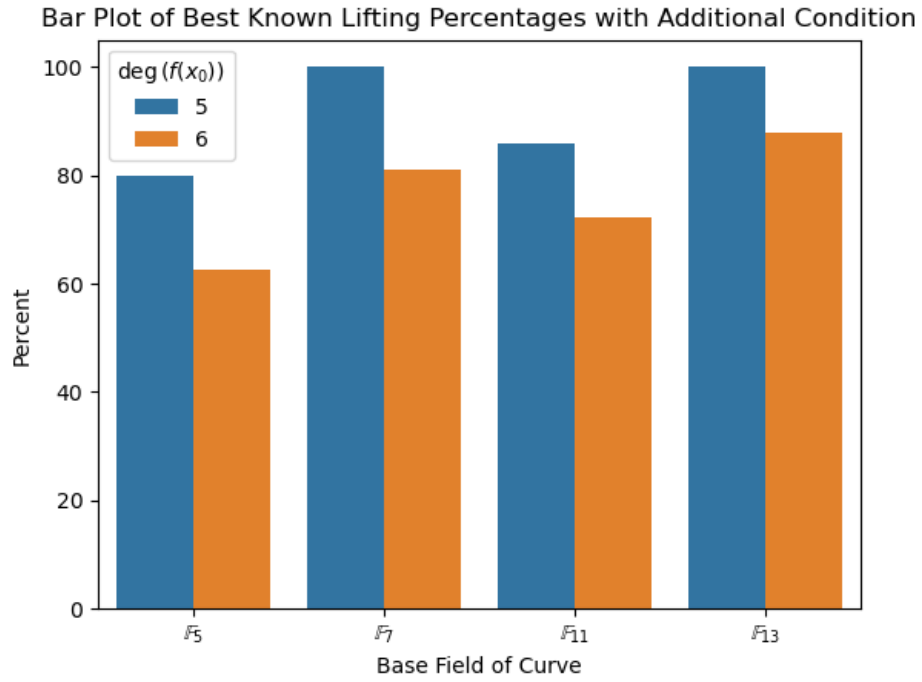


Figure 3.3: Bar plot of the percentage of hyperelliptic curves given by a monic polynomial $f(x_0)$ of degree five or six that *cannot* lift to Witt vectors of length three via the algorithm given that the coefficient on $x_0^{2(p-1)}$ of $f(x_0)^{(p-1)/2}$ is zero.

is a conjecture of Finotti, which has been proved for the second coordinate, and computationally proved for the third coordinate for primes less than or equal to 23.

3.4.2 Singular Affine Curves

We now turn our attention to curves that have the form of equation (3.1) but have singular points. Recall that this is equivalent to saying that $\gcd(f(x_0), f'(x_0)) \neq 1$, i.e., $f(x_0)$ has a repeated root. The theorems listed in Section 3.2 relied upon the curve being non-singular, so generally speaking those results do not hold. However, Algorithm 2 does not rely upon the curve being non-singular in order to attempt computing a lift. Since Theorem 3.2.5 does not apply, however, there is no guarantee that the best one can hope for any (now not hyperelliptic) lift is for all i

$$\deg F_i = \frac{dp^i - (d - 2)}{2}.$$

Despite this, if we are able to compute lifts of curves with small degree then this will be beneficial when exploring codes generated by these lifts in Chapter 4.

We attempted to use the algorithm to compute liftings of curves over $\mathbb{F}_3$, $\mathbb{F}_5$, and $\mathbb{F}_7$ of the same degree used in the previous section for comparison. We note that there are significantly less singular curves for a fixed degree d than there are hyperelliptic curves. The reason for this is due to the distribution of the discriminant of a random polynomial from $\mathbb{F}_q[x_0]$. Corollary 3.4 of (2) states that for any prime power q and integer $d \geq 2$, the number of monic polynomials in $\mathbb{F}_q[x_0]$ of degree d and discriminant 0 is q^{d-1} . Since the total number of monic polynomials is q^d , this means that there are $q^d - q^{d-1}$ many affine hyperelliptic curves while only q^{d-1} many singular curves.

It is worth asking how many of these monic polynomials meet the initial requirements to even attempt lifting, i.e., have a non-zero Hasse invariant and the equation $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ is integrable over $\mathbb{F}_q$. For the most part, these conditions are not satisfied by the majority of polynomials, whether singular or non-singular. Namely we found that 97.6% of the 817,566 polynomials examined

that resulted in hyperelliptic curves did not satisfy these first two conditions. For polynomials resulting in singular curves, 97.1% of the 150,980 polynomials examined did not satisfy the first two conditions.

For hyperelliptic curves we did not notice a pattern in what prevents these conditions from being satisfied. However, for singular curves, we found that the multiplicity of the roots of $f(x_0)$ affected whether a polynomial failed these first two conditions. Since the curve is singular, then $f(x_0)$ has at least one root of multiplicity at least two. We found, though, that if $f(x_0)$ has a root of multiplicity at least three over its splitting field and the degree of $f(x_0)^{(p-1)/2}$ is small enough, then it always failed one of these conditions. Namely, if the degree of $f(x_0)^{(p-1)/2}$ was less than $3p$, then either the Hasse invariant was zero, or the coefficient on x_0^{2p-1} or x_0^{3p-1} was non-zero, making $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ not integrable over $\mathbb{F}_p$. This leads to the following conjecture.

Conjecture 3.4.7. *Let $C/\mathbb{k}$ (where $\text{char}(\mathbb{k}) \geq 3$) be a singular curve given by equation (3.1) such that $f(x_0)$ has a root of multiplicity at least three over its splitting field and $\deg(f(x_0)^{(p-1)/2}) < 3p$. Then if the Hasse invariant is not 0, we have that $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ is not integrable over $\mathbb{k}$.*

While we were not able to prove the conjecture, we did verify it for all primes under 100. Note that if the Hasse invariant is non-zero and $\deg f(x_0)^{(p-1)/2} = 3p - 1$, then the leading term of $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ is $a_{3p-1}x_0^{3p-1}$ with $a_{3p-1} \neq 0$, so it is not integrable. For $\deg(f(x_0)^{(p-1)/2}) < 3p - 1$ and assuming A is non-zero, note the only coefficient of $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ that can make it not integrable is the coefficient on x_0^{2p-1} . We found that for all primes under 100, the Hasse invariant was a multiple of coefficient on x_0^{2p-1} . Hence, since $\mathbb{k}$ is a field, if the Hasse invariant is non-zero then the coefficient on x_0^{2p-1} is non-zero, causing the expression to be non-integrable.

While we currently have not found any conditions on multiplicity and higher degrees of $f(x_0)^{(p-1)/2}$, there could be such conditions. Note, though, that this implies

that the only singular curves for small degrees of $f(x_0)$ that the algorithm works with are those that have roots with multiplicity at most two. This puts a very strict condition on when there exists lifts of singular curves with a minimal degree on F_i in these cases.

For the remaining polynomials that did meet these initial conditions to lift via the algorithm, we examined if a lift achieving the lower bound for F_1 was able to do so for F_2 . Firstly, all polynomials $f(x_0)$ with a non-zero Hasse invariant where $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ was integrable were able to lift to a curve over $\mathbf{W}_2(\mathbb{F}_p)$ via the algorithm. Our analysis also showed that similarly to the case of hyperelliptic curves, if $A^{-1}f(x_0)^{(p-1)/2} - x_0^{p-1}$ was integrable over $\mathbb{F}_p$, then so was equation (3.9). Moreover, we found that for curves of genus two and three, all curves that the algorithm were able to lift to the second coordinate, were able to lift to the third coordinate.

This result was rather surprising, especially considering that over $\mathbb{F}_3$ in the hyperelliptic curve case for genus three most curves that were able to lift to the second coordinate via the algorithm were not able to lift to the third coordinate. Even more so, for $\mathbb{F}_5$ and $\mathbb{F}_7$ curves with genus two and three in the hyperelliptic case had over 10% of able to lift to the second coordinate but not the third.

We also note that the conditions in Proposition 3.4.5 and 3.4.6 do not apply in the case of singular curves. For example, consider the singular curve

$$C : / \mathbb{F}_5 : \quad y_0^2 = x_0^5 + x_0^2 + x_0 + 2,$$

which has $f(x_0)^2 = x_0^{10} + 2x_0^7 + \dots$ (so the coefficient of x_0^8 is 0), $a_0a_2 + a_1^2 = 3 \neq 0$ and lift

$$\begin{aligned} (y_0, G_1, G_2)^2 = & (1, 0, 0)(x_0, F_1, F_2)^5 + (0, 3, 1)(x_0, F_1, F_2)^4 + (0, 4, 1)(x_0, F_1, F_2)^3 \\ & + (1, 0, 0)(x_0, F_1, F_2)^2 + (1, 2, 0)(x_0, F_1, F_2) + (2, 2, 3) \end{aligned}$$

with

$$\begin{aligned}
F_1 &= x_0^{11} + 4x_0^8 + x_0^7 + 4x_0^6 + 3x_0^4 + 2x_0^2 + 4x_0, \\
F_2 &= x_0^{61} + 4x_0^{58} + x_0^{57} + 4x_0^{56} + 3x_0^{54} + 3x_0^{52} + 3x_0^{51} + 3x_0^{49} + x_0^{48} + 4x_0^{47} + x_0^{46} \\
&\quad + 3x_0^{43} + 3x_0^{42} + 4x_0^{41} + x_0^{40} + x_0^{38} + 3x_0^{36} + 3x_0^{34} + 2x_0^{33} + 4x_0^{32} + x_0^{31} \\
&\quad + x_0^{30} + x_0^{29} + 2x_0^{28} + 4x_0^{26} + 2x_0^{23} + 3x_0^{21} + x_0^{20} + x_0^{19} + 2x_0^{18} + 2x_0^{15} \\
&\quad + 3x_0^{14} + x_0^{13} + x_0^{12} + 2x_0^9 + 2x_0^8 + 4x_0^5 + 2x_0^4 + 3x_0^2 + x_0, \\
G_1 &= y_0 (2x_0^7 + 2x_0^6 + 3x_0^5 + 3x_0^4 + x_0^3 + 3x_0^2 + 2), \\
G_2 &= y_0 (3x_0^{90} + x_0^{87} + x_0^{86} + 2x_0^{85} + 3x_0^{84} + x_0^{83} + 2x_0^{81} + 2x_0^{80} + 3x_0^{75} + x_0^{72} + x_0^{71} \\
&\quad + x_0^{70} + 3x_0^{69} + x_0^{68} + 3x_0^{67} + 3x_0^{65} + 4x_0^{64} + 3x_0^{63} + x_0^{61} + 3x_0^{60} + x_0^{57} \\
&\quad + x_0^{56} + 2x_0^{55} + 4x_0^{51} + x_0^{50} + x_0^{49} + x_0^{48} + x_0^{47} + 2x_0^{46} + x_0^{45} + 3x_0^{43} + 4x_0^{42} \\
&\quad + 3x_0^{41} + 2x_0^{40} + x_0^{38} + 2x_0^{37} + 2x_0^{35} + x_0^{33} + 4x_0^{32} + 2x_0^{31} + x_0^{30} + x_0^{28} + 4x_0^{27} \\
&\quad + x_0^{26} + 4x_0^{25} + 4x_0^{24} + 2x_0^{21} + x_0^{20} + x_0^{19} + x_0^{18} + 2x_0^{17} + 3x_0^{16} + x_0^{15} + 3x_0^{13} \\
&\quad + 3x_0^{12} + 2x_0^{11} + x_0^{10} + x_0^9 + x_0^7 + 4x_0^6 + 2x_0^5 + 4x_0^4 + x_0^3 + x_0^2 + 2 * x_0 + 4)
\end{aligned}$$

We also have singular curve

$$C/\mathbb{F}_7 : \quad y_0^2 = x_0^5 + x_0 + 1,$$

which has $f(x_0)^3 = x_0^{15} + 3x_0^{11} + \dots$ (so the coefficient of x_0^{12} is 0), and lift

$$\begin{aligned}
(y_0, G_1, G_2) &= (1, 0, 0)(x_0, F_1, F_2)^5 + (0, 5, 5)(x_0, F_1, F_2)^4 + (0, 4, 0)(x_0, F_1, F_2)^3 \\
&\quad + (0, 2, 5)(x_0, F_1, F_2)^2 + (1, 4, 4)(x_0, F_1, F_2) + (1, 5, 5).
\end{aligned}$$

We omit the full equations for F_1, F_2, G_1 , and G_2 due to their length, as $\deg F_2 = 121$ and $\deg G_2 = 246$.

We make note of these curves here because when computing the generic matrix solutions that were used to computationally prove Proposition 3.4.5 and 3.4.6, we

found the discriminant of $f(x_0)$ appeared in the denominators of the equations for the lifted coefficients $\mathbf{a}_i$. For hyperelliptic curves, the discriminant is non-zero, which is not a problem. However, for singular curves the discriminant is zero, so it is interesting to note that a requirement for lifting hyperelliptic curves with best known minimal degree lifts does not apply to singular curves.

It was only until we examined curves of genus four that we found polynomials that could be lifted via the algorithm to the second coordinate but not the third. Over $\mathbb{F}_3[x_0]$, there were 486 polynomials with $d = 9$ that lifted to the second coordinate, and of those, 36 were not able to lift to the third coordinate. There were 486 polynomials with $d = 10$ that lifted to the second coordinate, and of those, 28 were not able to lift to the third coordinate.

With only a total of 64 singular curves that could lift via the algorithm to the second coordinate but not the third, there was not enough data to perform any kind of reliable interpolation to examine if any combination of the coefficients was indicative in the same manner for hyperelliptic curves. There was also no relationship among the placement of the roots, whether they were contained in $\mathbb{F}_p$ or some finite field extension $\mathbb{F}_q$.

These results show that while there are strict conditions when applying the lifting algorithm to singular curves, it still can be done. While we are uncertain if the degrees of F_i can be smaller, we do know that it is possible for singular curves to achieve the best known minimal degree lifts for hyperelliptic curves. However, as with hyperelliptic curves, we are not always guaranteed a lift where the degrees of F_i are minimized for all i if it can be minimized for F_1 . Further work can be done showing if lifts to the fourth coordinate with the algorithm are possible, and why the conditions that apply to hyperelliptic curves do not apply when the curve is singular.

Chapter 4

Applications to Coding Theory

We now discuss applications of the lifting algorithm described in Chapter 3 to coding theory. Before we can give concrete examples, we need to outline the basic theory of linear and affine linear codes. The elementary definitions can be found in (20) with additional detail. We also include a discussion on previous applications of lifts of hyperelliptic curves in the case of finite fields $\mathbb{F}_q$ with characteristic 2 from (8).

4.1 Linear Codes over Fields and Rings

A *code over an alphabet* Q in its most basic definition is a nonempty proper subset of Q^n , where Q is a set of distinct symbols. For linear codes, we assume that our alphabet is a finite field $\mathbb{k}$. Since $\mathbb{k}^n$ is a vector space, it makes sense to give a special definition to codes that are a linear subspace of $\mathbb{k}^n$.

Definition 4.1.1. Let $\mathbb{k}$ be a finite field of size $q = p^r$ for some prime p and $r \geq 1$. A q -ary linear code C is a subspace of $\mathbb{k}^n$. If C has dimension k , then C is called an $[n, k]$ code. Moreover, if G is a k by n matrix over $\mathbb{k}$ such that the rows are a basis for C , then G is called a *generator matrix* for C .

The elements of C are referred to as codewords. If $|C| = 1$, we refer to C as the trivial code. We also note that the generator matrix G is often presented in

standard form (or row reduced echelon form) $G = [I_k \ P]$ where I_k is the k by k identity matrix. We can use the generator matrix to determine when two codes are equivalent. If C_1 and C_2 are two codes with generator matrix G_1 and G_2 , respectively, and G_1 is equal to a permutation of the rows or columns of G_2 , then C_1 and C_2 are said to be equivalent codes.

One is often interested in error-correcting properties of codes. To that end, we introduce the idea of Hamming weight and Hamming distance.

Definition 4.1.2. Let C be a $[n, k]$ linear code over $\mathbb{k}$. For any $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{k}^n$, the *Hamming weight* $w_H(\mathbf{x})$ of $\mathbf{x}$ is defined as

$$w_H(\mathbf{x}) := |\{i : x_i \neq 0, 1 \leq i \leq n\}|.$$

For any $\mathbf{x}, \mathbf{y} \in \mathbb{k}^n$, the *Hamming distance* $d_H(\mathbf{x}, \mathbf{y})$ of $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$d_H(\mathbf{x}, \mathbf{y}) := w_H(\mathbf{x} - \mathbf{y}).$$

Note that the Hamming distance between two codewords is equivalent to counting the number of positions where $\mathbf{x}$ and $\mathbf{y}$ differ. The Hamming distance induces the discrete topology on the set $\mathbb{k}^n$ and can be used to measure the “closeness” of codewords. We define this more formally.

Definition 4.1.3. The *minimum Hamming weight* of a nontrivial code C is

$$\min\{w_H(\mathbf{x}) : \mathbf{x} \in C, \mathbf{x} \neq 0\}.$$

Similarly, the *minimum Hamming distance* of C is

$$\min\{d_H(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}.$$

For linear codes, it is widely known that the minimum distance is equivalent to the minimum weight of the non-zero codewords (as C is a linear subspace of $\mathbb{k}^n$, so for

all $\mathbf{x}, \mathbf{y} \in C$ we have $\mathbf{x} - \mathbf{y} \in C$). A linear code with minimum distance d is referred to as an $[n, k, d]$ code.

The process of encoding and decoding a code is as follows. First, a message (not necessarily an element of $\mathbb{k}^n$) is encoded into some codeword $\mathbf{x} \in C$. Then, $\mathbf{x}$ is transmitted over a communication channel where potential errors, $\mathbf{e}$, are added (if there are no errors, then $\mathbf{e} = (0, \dots, 0)$). The recipient of the received word $\mathbf{x} + \mathbf{e}$ then tries to detect the most likely $\mathbf{e}$, and decodes to some codeword $\mathbf{x}' \in C$ (which may or may not be equal to $\mathbf{x}$). Maximum-likelihood decoding assumes that one is more likely to receive fewer errors than more errors, and tries to correct any received word $\mathbf{y} \in \mathbb{k}^n$ to the codeword $\mathbf{x} \in C$ such that the distance $d_H(\mathbf{y}, \mathbf{x})$ is minimal, i.e., the codeword closest to $\mathbf{y}$ under the metric d_H .

Under maximum-likelihood decoding, an $[n, k, d]$ code where $d = 2e + 1$ is able to correct up to e errors in a received codeword. If $d = 2e$, then one can detect up to e errors, but is only able to correct up to $e - 1$ errors.

The concept of minimum distance allows us to interpret which codes are “good” and which codes are “bad”. As the size of the vector space n increases, there is more space to draw codewords from. As k increases, that means there are more codewords in C , which allows for more information to be communicated. As d increases, then one is able to correct more errors if the communication channel has high levels of noise.

Generally speaking, there is a trade-off between k and d . For C an $[n, k, d]$ code with generator matrix G in standard form and $\mathbf{x} \in C$, the first k positions correspond to the information symbols. Since C is a linear subspace of dimension k , the remaining $n - k$ positions of $\mathbf{x}$ are determined by G and are referred to as the *parity check symbols*. As k increases, the denseness of C in $\mathbb{k}^n$ increases, so the minimum distance between different codewords shrinks. However, the higher k is, the more distinct information one can transmit using the code.

For a code C to be a good code, there needs to be a high amount of information that can be transmitted (high k) with a decent minimum distance d for correction

purposes. If d is large, then C is sparse in the ambient space, which allows one to correct many errors, but also can be impractical due to the increased bandwidth required to transmit the code. If we have two codes C_1 and C_2 with the same ambient space $\mathbb{k}^n$ and C_2 allows a user to transmit more information (higher k) with a reasonable decrease in the minimum distance d , then C_2 is said to be a better code.

More results on linear codes can be found in (20), but this is all we need for our applications.

Linear codes have been extensively studied and there have been many books and papers written on the subject due to their nice properties coming from vector spaces. Recent advancements have been made into the study of affine linear codes, which are codes where the alphabet is no longer a field, but instead a ring. In (26), Walker provides a definition of linear codes over rings.

Definition 4.1.4. Let A be a ring. An (*affine*) *linear code* C of length n over A is a submodule of the free module A^n . If C is isomorphic to a free A -module, then C is a *free code* and the dimension of C is defined to be $\dim C = \text{rank}_A(C)$.

Walker connected affine linear codes and algebraic geometry through algebraic geometric codes. As in Section 3.2, let $\mathbf{C}/\mathbf{W}_\ell(\mathbb{k})$ be a projective curve over the ring of Witt vectors of length ℓ with reduction $C/\mathbb{k}$ modulo p . Let $\mathcal{P} := \{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n\}$ be a set of $\mathbf{W}_\ell(\mathbb{k})$ -rational points of $\mathbf{C}$ with distinct reductions modulo p , say $\{P_1, P_2, \dots, P_n\}$, D a Cartier divisor of $\mathbf{C}$ such that no $\mathcal{P}_i$ is in the support of D , and $\mathcal{L}(D) := \{f \in \overline{\mathbb{k}}(C)^* : \text{div}(f) \geq -D\} \cup \{0\}$, as defined at the end of Section 2.1. We then have the following definition and theorem from (26).

Definition 4.1.5. Let $\mathbf{C}$, $\mathcal{P}$, and $\mathcal{L}$ be as above. The algebraic geometric code over $\mathbf{W}_\ell(\mathbb{k})$ associated to $\mathbf{C}$, $\mathcal{P}$, and $\mathcal{L}$, denoted by $\mathcal{C}_{\mathbf{W}_\ell(\mathbb{k})}(\mathbf{C}, \mathcal{P}, \mathcal{L})$ as

$$\mathcal{C}_{\mathbf{W}_\ell(\mathbb{k})}(\mathbf{C}, \mathcal{P}, \mathcal{L}) := \{(\mathbf{f}(\mathbf{P}_1), \dots, \mathbf{f}(\mathbf{P}_n)) : \mathbf{f} \in \mathcal{L}\} \subset (\mathbf{W}_\ell(\mathbb{k}))^n.$$

Theorem 4.1.6 (Walker). *Let $\mathbf{C}$, $\mathcal{P} := \{\mathbf{P}_1, \dots, \mathbf{P}_n\}$, D , $\mathcal{L}$, and $\mathcal{C} := \mathcal{C}_{\mathbf{W}_\ell(\mathbb{k})}(\mathbf{C}, \mathcal{P}, \mathcal{L})$ be as above and let g denote the genus of the curve $\mathbf{C}$. If $(2g - 2) < \deg D < n$, then $\mathcal{C}$ is a linear code over $\mathbf{W}_\ell(\mathbb{k})$ which is free as a $\mathbf{W}_\ell(\mathbb{k})$ -module. Moreover, $\mathcal{C}$ has rank $\deg D + 1 - g$ and minimum Hamming distance at least $n - \deg D$.*

From this result comes the applications of the canonical lift and hyperelliptic lift to coding theory, which we will describe in the next section.

4.2 Previous Applications to Non-Linear Codes

In this section we discuss previous applications of minimal degree liftings (both for elliptic curves and hyperelliptic curves) over finite fields of characteristic $p = 2$. The ideas in this section are the ideas that we are replicating in Section 4.3.

Walker's result in (26) paved the way for Voloch and Walker to derive applications of canonical liftings of elliptic curves to coding theory in (23). They first developed their own algorithm to compute the canonical lift of an elliptic curve $E/\mathbb{k}$ to a curve $\mathbf{E}/\mathbf{W}_2(\mathbb{k})$, minimizing the degree of x_1 as mentioned in Section 3.1. Note that their derivation of dF_1/dx_0 is what is used in Algorithm 1.

To discuss their results, we must first define the Gray map. The *Gray map* $\phi_G : \mathbb{Z}/4\mathbb{Z} \rightarrow \mathbb{F}_2^2$ is defined by $\phi_G(0) = (0, 0)$, $\phi_G(1) = (0, 1)$, $\phi_G(2) = (1, 1)$, and $\phi_G(3) = (1, 0)$. We also define the *Lee distance*.

Definition 4.2.1. The *Lee weight* for $x \in \mathbb{Z}/m\mathbb{Z}$ with $m \in \mathbb{Z}_{\geq 1}$ is defined as

$$w_L(x) := \min \{x, m - x\}.$$

The *Lee metric* for $\mathbf{x} = (x_1, x_2, \dots, x_n) \in (\mathbb{Z}/m\mathbb{Z})^n$ is defined as

$$w_L(\mathbf{x}) := \sum_{i=1}^n w_L(x_i).$$

The *Lee distance* for $\mathbf{x}, \mathbf{y} \in (\mathbb{Z}/m\mathbb{Z})^n$ is defined as

$$d_L(\mathbf{x}, \mathbf{y}) := w_L(\mathbf{x} - \mathbf{y}).$$

We also define the minimum Lee distance in a similar fashion to the minimum Hamming distance. We note that the Gray map is an isometry from $(\mathbb{Z}/4\mathbb{Z}, d_L)$ to $(\mathbb{F}_2^2, d_H)$ where d_H is the Hamming distance as defined in Definition 4.1.2. Expanding the Gray map coordinate wise to $\Phi_G : (\mathbb{Z}/4\mathbb{Z})^n \rightarrow \mathbb{F}_2^{2n}$ provides a mapping of affine linear codes over $\mathbb{Z}/4\mathbb{Z}$ to non-linear binary codes. Thus producing codes over $\mathbb{Z}/4\mathbb{Z}$ with good minimum Lee distance results in good non-linear binary codes with regards to the Hamming distance.

Voloch and Walker found that by using exponential sums and minimizing the degree of the lifts of an elliptic curve $E/\mathbb{F}_{2^r}$ to an elliptic curve $\mathbf{E}/\mathbf{W}_2(\mathbb{F}_{2^r})$, they could generate an affine linear code C using Theorem 4.1.6 with good minimum Lee distance. They then applied the Gray map to the codewords of C to produce a binary non-linear code with good minimum Hamming distance.

Finotti was later able to extend their work by deriving a similar algorithm to compute minimal degree lifts to curves over $\mathbf{W}_3(\mathbb{F}_{2^r})$ in (8). He used these lifts to also generate non-linear binary codes with good minimum Hamming distance.

Up to this point, the only codes that have been generated are binary codes coming from curves over $\mathbb{F}_{2^r}$. This limitation was due to the fact that a Gray map from $\mathbb{Z}/p^\ell\mathbb{Z}$ for p prime and $p \neq 2$ had not been found until recently. While Finotti in (7) had an algorithm to compute minimal degree lifts in characteristic not two, the lack of a Gray map prevented generating good non-linear codes. However, due to recent advancements, we are now able to generate such codes. We go over these advancements in Section 4.3.

4.3 Modern Applications to Non-Linear Codes

Part of the difficulty in generalizing the Gray map was due to the nature of the Lee weight on $\mathbb{Z}/m\mathbb{Z}$ for $m \neq 2^r$. Carlet in (1) was able to derive a generalization of the Gray map using Boolean functions when $m = 2^r$, which was based on the Lee weight when $m = 4$. Carlet briefly defined a different Lee weight, when $m = 2^r$, but did not study it in depth. Later, Yildiz and Özger took this weight and expanded it to $m = p^r$ for any prime p in (27).

Definition 4.3.1. The *extended Lee weight* $w_L(x)$ of $x \in \mathbb{Z}_{p^\ell}$ is defined as

$$w_L(x) := \begin{cases} x & \text{if } x \leq p^{\ell-1}, \\ p^{\ell-1} & \text{if } p^{\ell-1} \leq x \leq (p-1)p^{\ell-1}, \\ p^\ell - x & \text{if } (p-1)p^{\ell-1} < x \leq p^\ell - 1 \end{cases}$$

Note that when $p^\ell = 4$, the extended Lee weight is equivalent to the Lee weight, and when $p = 2$, the weight matches the alternative definition derived by Carlet. We can extend this definition to a vector $\mathbf{x} \in (\mathbb{Z}/p^\ell\mathbb{Z})^n$ by taking the sum of the weights coordinate-wise, as was done for the Lee weight. This vector weight then gives way to the *extended Lee distance* d_L as in definition 4.1.2.

With this metric, Yildiz and Özger were able to define their own Gray map that is an isometry from $(\mathbb{Z}/p^\ell\mathbb{Z}, d_L)$ to $(\mathbb{F}_p^{p^{\ell-1}}, d_H)$ which they proved in (27).

Definition 4.3.2. For $x \in \mathbb{Z}/p^\ell\mathbb{Z}$ we define the *Gray map* $\phi_L : \mathbb{Z}/p^\ell\mathbb{Z} \rightarrow \mathbb{F}_p^{p^{\ell-1}}$ recursively in the following way. If $x \leq p^{\ell-1}$, let $\phi_L(x) = (1, \dots, 1, 0, \dots, 0)$, i.e., the vector with 1's in the first x coordinate positions and 0's in the remaining positions. If $x > p^{\ell-1}$, then there exists $q, r \in \mathbb{Z}_{\geq 0}$ such that $q \leq p-1$, $r \leq p^{\ell-1}-1$ and $x = qp^{\ell-1} + r$. Let $\phi_L(x) = \bar{q} + \phi_L(r)$ where $\bar{q} = (q, \dots, q)$ and addition is component-wise.

Theorem 4.3.3 (Yildiz, Özger). *The Gray map $\phi_L : (\mathbb{Z}/p^\ell\mathbb{Z}, d_L) \rightarrow (\mathbb{F}_p^{p^{\ell-1}}, d_H)$ is a distance preserving (not necessarily linear) map.*

Unfortunately, as noted by Carlet in (1), this type of weight does not have nice properties related to exponential sums. As such, constructing codes using best known minimal degree lifts of hyperelliptic curves and then applying the Gray map ϕ_L is not guaranteed to produce better (non-linear) codes with better minimum Hamming distance than simply constructing a code from any hyperelliptic lift, not necessarily minimal. Thus a different Gray map is required to produce good codes.

Progress with exponential sums was able to be made after relying on a different weight called the homogeneous weight introduced by Constantinescu and Heise in (3). It was further studied by Honold in (13), whose results were used by Voloch and Walker to give a formula for exponential sums in (24). They used these exponential sums to provide lower bounds for the minimum homogeneous weights of codes generated using Theorem 4.1.6. They acknowledged, though, that the results of Finotti in (6) (which we use) provided better bounds than their own.

We use their definition of homogeneous weight on the ring of Witt vectors over a finite field. Note that this definition is for extensions of finite fields $\mathbb{F}_q$ where $q = p^r$ for some prime p , but can be limited to $\mathbb{F}_p$ by replacing q with p in the definition.

Definition 4.3.4. The *homogeneous weight* $\text{wt}(\mathbf{x})$ of $\mathbf{x} \in \mathbf{W}_\ell(\mathbb{F}_q)$ is defined as

$$\text{wt}(x) := \begin{cases} 0 & \text{if } \mathbf{x} = \mathbf{0}, \\ q^{\ell-1} & \text{if } \mathbf{x} \in (p^{\ell-1}) \setminus \{\mathbf{0}\}, \\ (q-1)q^{\ell-2} & \text{otherwise.} \end{cases}$$

In the above definition, $(p^{\ell-1})$ refers to the ideal generated by the Witt vector $p^{\ell-1} = (0, \dots, 0, 1)$. As we saw in Section 2.2, we have that $\mathbf{W}_\ell(\mathbb{F}_p) \cong \mathbb{Z}/p^\ell\mathbb{Z}$ by the isomorphism φ . We can then extend the definition of homogeneous weight to $\mathbb{Z}/p^\ell\mathbb{Z}$ as found in (12).

Definition 4.3.5. The *homogeneous weight* $w_{\text{hom}}(x)$ of $x \in \mathbb{Z}/p^\ell\mathbb{Z}$ is defined as

$$w_{hom}(x) := \begin{cases} 0 & \text{if } x = 0, \\ p^{\ell-1} & \text{if } x \in (p^{\ell-1}) \setminus \{0\}, \\ (p-1)p^{\ell-2} & \text{otherwise.} \end{cases}$$

Note then for all $\mathbf{x} \in \mathbf{W}_\ell(\mathbb{F}_p)$ we have $\text{wt}(\mathbf{x}) = w_{hom}(\varphi(\mathbf{x}))$. Also note that when $p^\ell = 4$, we have the original Lee weight. We also define the homogeneous weight of a vector in $(\mathbb{Z}/p^\ell\mathbb{Z})^n$ and the *homogeneous distance* d_{hom} of two vectors in the same way as for the extended Lee distance.

While Voloch and Walker were able to apply the homogeneous weight to affine linear codes generated by curves, they did not produce a Gray map providing a way to generate non-linear codes from codes with good minimum homogeneous distance. This task was accomplished by Heng and Yue in (12) following the technique of Boolean functions used by Carlet in (1).

Definition 4.3.6. For $x \in \mathbb{Z}/p^\ell\mathbb{Z}$, let $x = \sum_{i=1}^{\ell} p^{i-1}x_i$ be its p -adic expansion, where $x_i \in \{0, 1, \dots, p-1\}$. Define the image $G(x)$ of x as

$$\begin{aligned} G(x) : \quad \mathbb{F}_p^{\ell-1} &\rightarrow \mathbb{F}_p \\ (y_1, \dots, y_{\ell-1}) &\mapsto x_\ell + \sum_{i=1}^{\ell-1} x_i y_i. \end{aligned}$$

Now let $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{p^{\ell-1}}\}$ be some ordering of the vectors of $\mathbb{F}_p^{p^{\ell-1}}$. Then the *generalized Gray map* is the map $G : \mathbb{Z}/p^\ell\mathbb{Z} \rightarrow \mathbb{F}_p^{p^{\ell-1}}$ where

$$x \mapsto (G(x)(\mathbf{y}_1), G(x)(\mathbf{y}_2), \dots, G(x)(\mathbf{y}_{p^{\ell-1}})).$$

We note that while it may appear that the Gray map is not well-defined due to the ordering of the vectors of $\mathbb{F}_p^{p^{\ell-1}}$ seeming arbitrary, recall that two linear codes are equivalent if one is simply a permutation of the rows or columns of the generating

matrix. As such, as long as the ordering of the vectors is fixed, the Gray map is well-defined. It also does not preserve linearity.

Theorem 4.3.7 (Heng, Yue). *The generalized Gray map G is an isometric embedding of $(\mathbb{Z}_{p^\ell}, d_{\text{hom}})$ into $(\mathbb{F}_p^{p^\ell-1}, d_H)$.*

This theorem allows one to generate non-linear codes with good minimum Hamming distance over $\mathbb{F}_p$ from linear codes with good minimum homogeneous distance. Thus one can use the results of Finotti to create best known minimal degree lifts of hyperelliptic curves, the results of Voloch and Walker to generate an affine linear code with good minimum homogeneous distance, and the results of Heng and Yue to produce non-linear codes with good minimum Hamming distance. We discuss the process to do so in the next section.

4.4 Algorithms

In this section we include all the necessary algorithms required to construct a linear code over $\mathbf{W}_\ell(\mathbb{F}_p)$ from a hyperelliptic lift following Algorithm 2. We note that in principal, Algorithm 2 does allow one to lift curves over $\mathbb{F}_q$ for $q = p^r$ with $r > 1$. This would result in a code over $\mathbf{W}_\ell(\mathbb{F}_q)$, which one can apply the trace map tr as given in equation (2.10) to then give a code over $\mathbf{W}_\ell(\mathbb{F}_p)$. However, the techniques described in the previous section to increase the lower bound of the minimum homogeneous distance do not apply, as for a vector $\mathbf{x} \in \mathbf{W}_\ell(\mathbb{F}_q)$ one does not have that $\text{wt}(\mathbf{x}) = w_{\text{hom}}(\text{tr}(\mathbf{x}))$ (in fact, if $q \neq p$, then it is only equal for the zero vector). So we restrict ourselves to examining curves over $\mathbb{F}_p$. This ensures that using the best known minimal degree lift will result in better codes.

Before we discuss the details, we give an overview of the process. Given a hyperelliptic curve C over $\mathbb{F}_p$, we first use Algorithm 1 or 2 to generate a lift of the curve to a curve $\mathbf{C}$ over $\mathbf{W}_\ell(\mathbb{F}_p)$, where $\ell = 2, 3$. To use Theorem 4.1.6 properly, we need a collection of points $\mathcal{P} = \{\mathbf{P}_1, \dots, \mathbf{P}_n\} \subset \mathbf{W}_\ell(\mathbb{F}_p)(\mathbf{C})$ and a divisor $D \in \text{Div}(\mathbf{C})$

such that no $\mathbf{P}_i$ is in the support of D . The length of the code depends on the number of points, so we want $\mathcal{P}$ to be larger so that we have a longer code. To that end, since our lift is of the affine portion of the curve, we choose

$$D = \begin{cases} n\mathcal{O}, & \text{if } \deg(f) \text{ is odd,} \\ n(\mathcal{O}_1 + \mathcal{O}_2), & \text{if } \deg(f) \text{ is even,} \end{cases}$$

with $\{\mathcal{O}\}$ and $\{\mathcal{O}_1, \mathcal{O}_2\}$ the respective set of points at infinity, and $n \geq 1$. This means that we can use any subset of the affine rational points on our curve $\mathbf{C}$. We can obtain such a subset by taking all (distinct) affine $\mathbb{F}_p$ -rational points (α, β) on C and lifting them to (distinct) $\mathbf{W}_\ell(\mathbb{F}_p)$ -rational points $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ on $\mathbf{C}$. Note that Theorem 4.1.6 gives us bounds on n depending on the genus of the curve and the number of points in $\mathcal{P}$ to ensure that the resulting code is linear over $\mathbf{W}_\ell(\mathbb{F}_p)$.

After generating $\mathcal{P}$, we need to create the generating matrix for the code. To do this, we will have to generate a basis for $\mathcal{L}(D)$. The functions in $\mathcal{L}(D)$ we can view as lifts of functions contained in $\mathcal{L}(D')$, where $D' \in \text{Div}(C)$ is the reduction modulo p of D . The function field $\overline{\mathbb{F}_p}(C)^*$ consists of functions in terms of x_0 and y_0 , so we will use these to generate our basis. Recall that $g \in \mathcal{L} \iff \text{div}(g) \geq -D'$. In other words, a function $g(x_0, y_0)$ will be in $\mathcal{L}$ if the absolute value of the order of g at $\mathcal{O}$ is less than $n_{\mathcal{O}}$. Thus the basis of $\mathcal{L}$ depends on the order of x_0 and y_0 at $\mathcal{O}$.

Our basis will then consist of increasing powers and products of x_0 and y_0 based on their orders at the point (or points) at infinity. For example, the basis for $\mathcal{L}(7\mathcal{O})$ where $d = 5$ is $\{1, x_0, x_0^2, y_0, x_0^3, y_0x_0\}$.

Once we have our basis, say $\{\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_k\}$, then the rows of the generating matrix are formed by evaluating the basis functions at the lifted points.

$$\begin{bmatrix} \mathbf{f}_1(\mathbf{P}_1) & \mathbf{f}_1(\mathbf{P}_2) & \cdots & \mathbf{f}_1(\mathbf{P}_n) \\ \mathbf{f}_2(\mathbf{P}_1) & \mathbf{f}_2(\mathbf{P}_2) & \cdots & \mathbf{f}_2(\mathbf{P}_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{f}_k(\mathbf{P}_1) & \mathbf{f}_k(\mathbf{P}_2) & \cdots & \mathbf{f}_k(\mathbf{P}_n) \end{bmatrix}$$

Next, we convert the code from being over $\mathbf{W}_\ell(\mathbb{F}_p)$ to being over $\mathbb{Z}/p^\ell\mathbb{Z}$ by applying the isomorphism φ to the entries of the matrix. We then perform a pseudo-row reduction algorithm to put the matrix into a reduced form (note that since $\mathbb{Z}/p^\ell\mathbb{Z}$ is a ring and not a field, we cannot perform Gaussian elimination in the typical way). To generate the codewords of the code, we take all vectors of the free module $(\mathbb{Z}/p^\ell\mathbb{Z})^k$ and multiply them on the left of the generating matrix in the typical way. Finally, to produce the non-linear code we apply either Gray map described in the previous section to the coordinates of each codeword, producing codewords over $\mathbb{F}_p$ of length $np^{\ell-1}$.

The first algorithm that we provide is that to generate the lifted points on the curve over the ring of Witt vectors. We assume that the output of Algorithm 1 or 2 has been stored prior to implementing this algorithm. We note that SageMath has built in elliptic curve and hyperelliptic curve objects, which can be utilized to quickly get all rational points on the curve to lift. The points are presented in projective form, so we have to remove the point at infinity and the projective coordinate. Additionally, while hyperelliptic curves with d even have two points at infinity, SageMath only includes one point at infinity due to the choice of model.

In principle, one does not have to select all the rational points on the curve. The algorithm can be modified to only select a certain number of affine points, resulting in a smaller code. This algorithm though selects all affine points on the curve. In

Algorithm 3, HyperellipticCurve is a hyperelliptic curve object and CurveLift is the output of Algorithm 1 or 2.

Algorithm 3 Lift rational points on $\mathbb{F}_p$ to points on $\mathbf{W}_\ell(\mathbb{F}_p)$

```

1: procedure LIFT_POINTS(HyperellipticCurve, CurveLift)
2:   pts  $\leftarrow$  list of projective points from HyperellipticCurve
3:   for  $i \leftarrow 0, \text{len}(\text{pts})$  do
4:     if pts[i][2] = 0 then
5:       delete pts[i] ▷ Remove point at infinity
6:       break
7:   lifted_pts  $\leftarrow$  [] ▷ List of lifted Witt vector points
8:   x_cor  $\leftarrow$  [] ▷ Witt vector of  $x$ -coordinate of lifted point
9:   y_cor  $\leftarrow$  [] ▷ Witt vector of  $y$ -coordinate of lifted point
10:  if len(CurveLift) = 4 then
11:    curve_type = 0 ▷ Ordinary elliptic curve
12:  else
13:    curve_type = 1 ▷ Hyperelliptic curve
14:  for  $i \leftarrow 0, \text{len}(\text{pts})$  do
15:    for  $j \leftarrow 0, \text{len}(\text{CurveLift}[1])$  do ▷ Length of Witt vector
16:      x_cor[j]  $\leftarrow$  CurveLift[1 + curve_type][j](pts[i][0])
17:      y_cor[j]  $\leftarrow$  CurveLift[2 + curve_type][j](pts[i][0])  $\cdot$  (pts[i][1])
18:    append [x_cor, y_cor] to lifted_pts
  return lifted_pts

```

Algorithm 4 takes the list of lifted points, the degree d of $f(x_0)$, and the degree of the divisor (i.e., n for hyperelliptic curves with d odd and $2n$ for d even) and produces the generating matrix. We make use of several functions for Witt vectors that are based of Finotti's code in Magma. Namely we use WITTPOWER($\mathbf{x}, n$), which raises a Witt vector $\mathbf{x}$ to the power n , and WITTPROD($\mathbf{x}, \mathbf{y}$) which computes the product $\mathbf{x} \cdot \mathbf{y}$. The details of these functions are omitted, and can be found in `witt.sage` (see Section 3.3). The degree d determines the construction of the basis as that determines the order of x_0 and y_0 at the point (or points) at infinity.

The difficulty of this algorithm is generating the basis for $\mathcal{L}$ depending on the degree of the divisor D . For elliptic curves, this is straightforward as $\text{ord}_{\mathcal{O}}(x_0) = -2$ and $\text{ord}_{\mathcal{O}}(y_0) = -3$. Thus an increase in the degree of the divisor either adds a power

of x_0 to the basis, or a power of x_0 times y_0 . For example, the basis (in increasing degree) for a divisor of degree seven would be $\{1, x_0, y_0, x_0^2, x_0y_0, x_0^3, x_0^2y_0\}$.

For hyperelliptic curves we must be more careful. If d is odd, then $\text{ord}_{\mathcal{O}}(x_0) = -2$ and $\text{ord}_{\mathcal{O}}(y_0) = -d$. Then y_0 does not appear in our basis until $n \geq d$ (where again $D = n\mathcal{O}$). We choose our basis to consist of powers of x_0 and the product of a power of x_0 times y_0 , as in the case for elliptic curves. For example, the basis (in increasing degree) for a divisor of degree seven with $d = 5$ would be $\{1, x_0, x_0^2, y_0, x_0^3, x_0y_0\}$.

If d is even, then we have two points at infinity, $\mathcal{O}_1$ and $\mathcal{O}_2$. We assume for our algorithm that our divisor is $D = n(\mathcal{O}_1 + \mathcal{O}_2)$. We have that $\text{ord}_{\mathcal{O}_i}(x_0) = -1$ and $\text{ord}_{\mathcal{O}_i}(y_0) = -d/2$, so our choice of divisor ensures our basis functions are still made of powers and products of x_0 and y_0 . Note though that when $\deg D \geq d$, then we have a choice as to whether we include $x_0^{d/2}$ in the basis or y_0 . By convention, our algorithm includes y_0 .

Once we have our basis, we form the generating matrix by evaluating each of the lifted points at the basis functions.

The output G of Algorithm 4 is a matrix with elements in $\mathbf{W}_\ell(\mathbb{F}_p)$. This gives us a generating matrix for an affine linear code, which we can convert to a code over $\mathbb{Z}/p^\ell\mathbb{Z}$ by applying the isomorphism in Section 2.2 to the entries of G . The code to do so in SageMath is included in the file `isom.sage` and is omitted here for sake of brevity. Since $\mathbb{Z}_p$ is implemented in SageMath along with the Teichmüller representatives, the algorithm is fairly straightforward.

The last algorithm that we include is the algorithm to take the matrix G over $\mathbb{Z}/p^\ell\mathbb{Z}$ and put it in standard form. We note that since we are concerned only with the code resulting from the generating matrix, we can permute the rows and columns in any way that we wish as it will result in an equivalent code.

Since $\mathbb{Z}/p^\ell\mathbb{Z}$ is a ring with zero divisors, we have to express some caution when performing row operations. We start at the top left-most position of G and search the first row and first column for a unit in $\mathbb{Z}/p^\ell\mathbb{Z}$. Once we find a unit, we can perform a row or column swap to move it into the top left position and multiply the row

Algorithm 4 Generate the generating matrix for an affine linear code from a hyperelliptic lift

```

1: procedure GENERATE_MATRIX(lifted_pts, d, dvsr_deg)
2:    $n \leftarrow \text{len}(\text{lifted\_pts})$  ▷ Number of affine points
3:    $\ell \leftarrow \text{len}(\text{lifted\_pts}[0][0])$  ▷ Length of Witt vectors
4:    $G \leftarrow []$  ▷ Initialize the generator matrix
5:   for  $j \leftarrow 0, n - 1$  do ▷ First row of  $G$  from constant function
6:      $v \leftarrow [1]$ 
7:     for  $i \leftarrow 1, \ell$  do ▷ Witt vector 1 of length  $\ell$ 
8:        $v[i] \leftarrow 0$ 
9:      $G[0][j] \leftarrow v$ 
10:  if  $\ell = 3$  then ▷ Elliptic curve case
11:     $\text{pwr\_x} \leftarrow 1$ 
12:    for  $i \leftarrow 2, \text{dvsr\_deg}$  do
13:       $\text{next\_row} \leftarrow []$ 
14:      if  $i$  is even then
15:        for  $j \leftarrow 0, n - 1$  do
16:          append WITTPOWER( $\text{lifted\_pts}[j][0], \text{pwr\_x}$ ) to  $\text{next\_row}$ 
17:         $\text{pwr\_x} \leftarrow \text{pwr\_x} - 1$ 
18:      else
19:        for  $j \leftarrow 0, n - 1$  do
20:          if  $i > 1$  then
21:             $x \leftarrow \text{WITTPOWER}(\text{lifted\_pts}[j][0], \text{pwr\_x})$ 
22:             $y \leftarrow \text{WITTPROD}(x, \text{lifted\_pts}[j][1])$ 
23:            append  $y$  to  $\text{next\_row}$ 
24:          else
25:            append  $\text{lifted\_pts}[j][1]$  to  $\text{next\_row}$ 
26:           $\text{pwr\_x} \leftarrow \text{pwr\_x} + 2$ 
27:           $G[i - 1] \leftarrow \text{next\_row}$ 

```

Algorithm 4 GENERATE_MATRIX continued

```
28:   else ▷ Hyperelliptic curve case
29:     basis  $\leftarrow [(1, 0)]$ 
30:     if  $d$  is even then
31:       while  $2 \cdot \text{basis}[-1][0] + d \cdot \text{basis}[-1][1] \leq \text{dv\_sr\_deg}$  do
32:         if  $\text{basis}[-1][0] < (d - 2)/2$  then
33:           append ( $\text{basis}[-1][0] + 1, \text{basis}[-1][1]$ ) to basis
34:         else
35:           append ( $0, \text{basis}[-1][1] + 1$ ) to basis
36:     else
37:       while  $2 \cdot \text{basis}[-1][0] + d \cdot \text{basis}[-1][1] \leq \text{dv\_sr\_deg}$  do
38:         if  $2 \cdot \text{basis}[-1][0] + d \cdot \text{basis}[-1][1] < d - 1$  then
39:           append ( $\text{basis}[-1][0] + 1, 0$ ) to basis
40:         else
41:           if  $\text{basis}[-1][1] = 0$  then
42:             if  $\text{basis}[-1][0] = (d - 1)/2$  then
43:               append ( $0, 1$ ) to basis
44:             else
45:               append ( $\text{basis}[-2][0] + 1, 0$ ) to basis
46:           else
47:             append ( $\text{basis}[-2][0] + 1, 0$ ) to basis
48:   Delete basis[-1] ▷ Remove last entry that has degree just past dv\_sr\_deg
49:   for  $i \leftarrow 0, \text{len}(\text{basis}) - 1$  do
50:     next_row  $\leftarrow []$ 
51:     if  $\text{basis}[i][0] = 0$  or  $\text{basis}[i][1] = 0$  then
52:       if  $\text{basis}[i][1] = 0$  then
53:         for  $j \leftarrow 0, n - 1$  do
54:            $x \leftarrow \text{WITTPower}(\text{lifted\_pts}[j][0], \text{basis}[i][0])$ 
55:           append  $x$  to next_row
56:       else
57:         for  $j \leftarrow 0, n - 1$  do
58:            $y \leftarrow \text{WITTPower}(\text{lifted\_pts}[j][1], \text{basis}[i][1])$ 
59:           append  $y$  to next_row
60:     else
61:       for  $j \leftarrow 0, n - 1$  do
62:          $x \leftarrow \text{WITTPower}(\text{lifted\_pts}[j][0], \text{basis}[i][0])$ 
63:          $y \leftarrow \text{WITTPower}(\text{lifted\_pts}[j][1], \text{basis}[i][1])$ 
64:          $z \leftarrow \text{WITTPROD}(x, y)$ 
65:         append  $z$  to next_row
66:    $G[i + 1] \leftarrow \text{next\_row}$ 
return  $G$ 
```

by the inverse of the entry to turn it into a leading one. We then perform normal row operations to make all entries under the leading one become zero. Afterwards, we move diagonally down the matrix, repeating the process. Once we have created a leading one, that row and column are considered reduced and we reduce the remaining submatrix.

Note that if we do not find a unit in the first row or first column, we can search the whole matrix until we find one. Once we find one we can perform necessary row or column swaps to move it into the leading position. If there are enough units in the matrix, we can produce an upper triangular matrix with ones along the diagonal.

If there are not enough units to get a diagonal of all ones, we then resort to expressing the diagonal as powers of p in increasing order. First we use p . To do so we search the remaining submatrix for a nonzero element of the ideal (p) but not the ideal (p^2) . Suppose then we find $\alpha \in (p) \setminus (p^2)$. Then $\alpha = up$ for some unit $u \in (\mathbb{Z}/p^\ell\mathbb{Z})^*$. Let $\beta = u^{-1}(p^{\ell-1} + 1) \bmod p^\ell$. Then $\alpha\beta \equiv p \bmod p^\ell$. Thus we can multiply the row with α in the leading position by β to reduce the row. We can continue this trend with increasing powers of p as necessary. For our purposes, our codes are either over $\mathbb{Z}/p^2\mathbb{Z}$ or $\mathbb{Z}/p^3\mathbb{Z}$ so our reduced matrices will have 1, p , or p^2 along the diagonal.

The pseudocode for this algorithm is included in Algorithm 5. There are several auxiliary functions that are used to find units, elements of the ideal (p) and (p^2) , and reduce the entries in a leading column to zero. We describe their inputs and output here, but more details can be found in the file `ReducedForm.sage`. The function `FINDUNIT` searches the given row and column of a matrix and tries to find a unit. The first Boolean value it returns indicates if a unit has been found, the second Boolean indicates if the unit was found in the given row (True) or column (False), and finally it returns the corresponding rows or columns needed to swap to move the unit to the leading position. The function `FINDNONUNITIDEAL` does the same, only searching for an element in the ideal generated by the prime power p^r given that is

not in the ideal generated by prime power p^{r+1} . The details for these algorithms are omitted and can be found in the file `ReducedForm.sage`.

Once the generating matrix G has been put in standard form, computing the minimum homogeneous distance and minimum extended Lee distance is fairly straightforward. The rows of G form a basis for a free module C that is our code, so finding the minimum distance is equivalent to computing the minimum weight of all codewords in C . This is due to the fact that if $\mathbf{x}, \mathbf{y} \in C$, then $\mathbf{x} - \mathbf{y} \in C$.

Unfortunately, there are no known algorithms for efficient computation of the minimum distance of a code given a generating matrix in polynomial time. Much research has been done in studying the minimum Hamming distance for linear codes over finite fields, especially for binary codes. Vardy did show in (21) that computing the minimum Hamming distance for a binary linear code is NP-hard, and the corresponding decision problem is NP-complete. So while one does not have a polynomial time algorithm for solving the minimum distance of a given code, one can verify if a hypothesized minimum distance is correct in polynomial time. Dumer showed in (5) that the minimum distance is not approximable to within any constant factor in random polynomial time (RP).

For weights over affine linear codes there has not been as much research. As such, the best algorithm that we have written to compute the minimum homogeneous distance and minimum extended Lee distance of an affine linear code is by brute force. We first compute all codewords in C , compute the weights of all non-zero codewords, and then take the minimum value. We omit the details of these algorithms here as they contain no computational insight, but they can be found in the file `MinimumDistance.sage`.

The last algorithm that we have written computes the image of a vector under the Gray maps ϕ_L and G . However, due to Theorem 4.3.3 and 4.3.7, we only need compute the minimum extended Lee distance and minimum homogeneous distance, respectively, to find the minimum Hamming distance of the resulting non-linear code.

Algorithm 5 Put a matrix over $\mathbb{Z}/p^\ell\mathbb{Z}$ in standard form

```

1: procedure STANDARD_FORM( $G, p, \ell$ )
2:   numRow  $\leftarrow$  len( $G$ )
3:   numCol  $\leftarrow$  len( $G[0]$ )
4:   sorted  $\leftarrow$  False ▷ Track if diagonal fully sorted
5:    $i \leftarrow 0$  ▷ Track row of leading position
6:    $j \leftarrow 0$  ▷ Track column of leading position
7:    $k \leftarrow 1$  ▷ Track how many rows/columns searched
8:   SortUnits  $\leftarrow$  True
9:   while SortUnits do
10:    UnitInfo  $\leftarrow$  FINDUNIT( $G, i, j$ )
11:    if UnitInfo[0] then ▷ If a unit is found
12:      if UnitInfo[1] then
13:        Swap matrix columns UnitInfo[2] and UnitInfo[3]
14:      else
15:        Swap matrix rows UnitInfo[2] and UnitInfo[3]
16:        Make leading position 1 and reduce column  $j$ 
17:         $i \leftarrow i + 1$ 
18:         $j \leftarrow j + 1$ 
19:      else
20:        Move row  $i$  to the bottom of the matrix
21:         $k \leftarrow k + 1$ 
22:        SortUnits  $\leftarrow (k \leq \text{numRow})$  and  $(k \leq \text{numCol})$ 
23:    if  $i \geq \text{numRow}$  or  $j \geq \text{numCol}$  then ▷ Check diagonal all sorted
24:      sorted  $\leftarrow$  True
25:    if not sorted and  $\ell \geq 2$  then
26:      SortIdealP  $\leftarrow$  True
27:       $k \leftarrow i + 1$  ▷ Start from first non-unit position
28:      while SortIdealP do
29:        NonUnitInfo  $\leftarrow$  FINDNONUNITIDEAL( $G, p, i, j$ )
30:        if NonUnitInfo[0] then
31:          if NonUnitInfo[1] then
32:            Swap matrix columns NonUnitInfo[2] and NonUnitInfo[3]
33:          else
34:            Swap matrix rows NonUnitInfo[2] and NonUnitInfo[3]
35:            Make leading position  $p$  and reduce column  $j$ 
36:             $i \leftarrow i + 1$ 
37:             $j \leftarrow j + 1$ 
38:          else
39:            Move row  $i$  to the bottom of the matrix

```

Algorithm 5 STANDARD_FORM continued

```
40:          $k \leftarrow k + 1$ 
41:         SortIdealP  $\leftarrow (k \leq \text{numRow})$  and  $(k \leq \text{numCol})$ 
42:     if  $i \geq \text{numRow}$  or  $j \geq \text{numCol}$  then
43:         sorted  $\leftarrow \text{True}$ 
44:     if not sorted and  $\ell = 3$  then
45:         SortIdealPSquared  $\leftarrow \text{True}$ 
46:          $k \leftarrow i + 1$ 
47:         while SortIdealPSquared do
48:             NonUnitInfo  $\leftarrow \text{FINDNONUNITIDEAL}(G, p^2, i, j)$ 
49:             if NonUnitInfo[0] then
50:                 if NonUnitInfo[1] then
51:                     Swap matrix columns NonUnitInfo[2] and NonUnitInfo[3]
52:                 else
53:                     Swap matrix rows NonUnitInfo[2] and NonUnitInfo[3]
54:                     Make leading position  $p$  and reduce column  $j$ 
55:                      $i \leftarrow i + 1$ 
56:                      $j \leftarrow j + 1$ 
57:                 else
58:                     Move row  $i$  to the bottom of the matrix
59:                      $k \leftarrow k + 1$ 
60:                     SortIdealPSquared  $\leftarrow (k \leq \text{numRow})$  and  $(k \leq \text{numCol})$ 
61:     if  $i < \text{numRow}$  then
62:         Delete any trailing rows of zeros
63:     else if  $j < \text{numCol}$  then
64:         Delete any trailing columns of zeros
return  $G$ 
```

For this reason we omit the details of these algorithms as well, but they may be found in the file `graymap.sage`.

4.5 Examples and Results

In this section we discuss our exploratory analysis of non-linear codes generated using the algorithms described in the previous section. To generate these results we used hyperelliptic curves over the finite fields $\mathbb{F}_3$, $\mathbb{F}_5$, and $\mathbb{F}_7$. We will denote the hyperelliptic curve over $\mathbb{F}_p$ by C , its absolute minimal degree lift over the ring of Witt vectors of length ℓ (if it exists) by $\mathbf{C}$, and the resulting code over the ring $\mathbb{Z}/p^\ell\mathbb{Z}$ by $\mathcal{C}$.

Of ultimate interest are the resulting non-linear codes after applying the Gray maps ϕ_L and G . Note that only the minimum homogeneous distance is benefited by minimizing the degree of F_i of the hyperelliptic lift of C . As such, one would hope that the code $\mathcal{C}$ has a greater minimum homogeneous distance than minimum extended Lee distance, so that the resulting non-linear code will have greater minimum Hamming distance.

We will also use the information rate of a code to determine what codes are good codes. For a code C from alphabet Q where $|Q| = b$ and the length of C is n has *information rate*

$$R := \frac{\log_b(|C|)}{n}.$$

Before we list examples, let's consider the change in information rate after applying the Gray map. For a code C over $\mathbb{Z}/p^\ell\mathbb{Z}$ of length n , we have that the information rate of the affine linear code (denoted R_L) is

$$R_L = \frac{\log_{p^\ell}(|C|)}{n}.$$

Both Gray maps ϕ_L and G are injective, so $|C|$ will stay the same after applying either map. The alphabet of the code though will change from $\mathbb{Z}/p^\ell\mathbb{Z}$ to $\mathbb{F}_p$, and the code

length will change to $np^{\ell-1}$. Thus the information rate for the resulting non-linear code (denoted R_N) is

$$R_N = \frac{\log_p(|C|)}{np^{\ell-1}}.$$

Applying a change of log base, we can compare R_N and R_L and note that

$$R_N = \frac{\ell}{p^{\ell-1}} R_L.$$

So, our information rate will decrease by a factor of $\ell/(p^{\ell-1})$.

The first thing to note is that for all computations, the minimum homogeneous distance was higher than that of the minimum extended Lee distance, often by a significant margin. For example, consider the hyperelliptic curve

$$C/\mathbb{F}_5 : \quad y_0^2 = x_0^6 + 4x_0^2 + 4,$$

which is lifted to the curve $\mathcal{G}(\mathbf{C})$

$$\begin{aligned} (y_0, G_1, G_2)^2 = & (1, 0, 0)(x_0, F_1, F_2)^5 + (0, 0, 3)(x_0, F_1, F_2)^4 + (0, 2, 3)(x_0, F_1, F_2)^3 \\ & + (4, 4, 3)(x_0, F_1, F_2)^2 + (2, 4, 1)(x_0, F_1, F_2) + (4, 4, 3), \end{aligned}$$

with

$$\begin{aligned} F_1 &= x_0^{11} + x_0^8 + 2x_0^7 + 3x_0^6 + 4x_0^4 + 2x_0^3 + 3x_0^2 + x_0, \\ G_1 &= y_0(3x_0^7 + 4x_0^6 + 3x_0^5 + 3x_0^4 + 3x_0^3 + 3x_0^2 + 3x_0 + 3). \end{aligned}$$

(We omit F_2 and G_2 due to their length). Using the divisor $D = 3(\mathcal{O})$ and basis $\{1, x_0\}$ for $\mathcal{L}(D)$, we have the resulting generator matrix G over $\mathbb{Z}/5^3\mathbb{Z}$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 9 & 9 & 53 & 53 & 17 \\ 0 & 1 & 0 & 1 & 117 & 117 & 73 & 73 & 109 \end{bmatrix}.$$

The resulting code $\mathcal{C}$ has length 9 and 5^6 codewords with minimum homogeneous distance 160 and minimum extended Lee distance 9.

It is startling to note how much minimizing the degree of F_i and using the homogeneous weight is an improvement to the minimum distance of the corresponding code instead of the extended Lee weight under the respective Gray maps.

We note, though that this would not be considered a good code. This minimum homogeneous distance is substantially higher than it needs to be. This means that the resulting code after applying the Gray map G would be a code of length $9 \cdot 5^2 = 405$, and while one would be able to detect up to 80 errors and correct 79 of them with maximum-likelihood decoding, this code has an information rate of only 0.027.

We can improve the information rate by increasing the degree of D , which will increase the dimension of $\mathcal{L}$ and the number of rows of G . Letting $D = 4(\mathcal{O})$ we have basis $\{1, x_0, x_0^2\}$ and generating matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 21 & 21 & 108 \\ 0 & 1 & 0 & 1 & 0 & 0 & 32 & 32 & 42 \\ 0 & 0 & 1 & 0 & 0 & 1 & 73 & 73 & 101 \end{bmatrix},$$

which results in a code with 5^9 codewords and minimum homogeneous distance 100. Applying the Gray map still results in a code of length 405, but with the increased number of codewords the information rate as a non-linear code over $\mathbb{F}_5$ is now 0.04.

To improve the information rate we can choose a lift to Witt vectors of length two, reducing $\ell/(p^{\ell-1})$ and thereby reducing the amount the information rate is decreased after applying the Gray map. For example, consider the ordinary elliptic curve

$$E/\mathbb{F}_3 : y_0^2 = x_0^3 + x_0^2 + 1$$

that has lift $\mathcal{G}(\mathbf{E})$

$$(y_0, y_1) = (1, 0)(x_0, x_1)^3 + (1, 0)(x_0, x_1)^2 + (1, 0)(x_0, x_1) + (1, 2),$$

where

$$\begin{aligned}x_1 &= x_0^4 + 2x_0^2 + x_0, \\y_1 &= y_0(x_0^2 + x_0 + 1).\end{aligned}$$

Letting $D = 3(\mathcal{O})$ gives the basis $\{1, x_0, y_0\}$ of $\mathcal{L}$ and generating matrix (over $\mathbb{Z}/3^2\mathbb{Z}$)

$$\begin{bmatrix} 1 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 & 8 \\ 0 & 0 & 1 & 7 & 2 \end{bmatrix}.$$

The resulting code has 3^6 codewords with minimum homogeneous distance four. The information rate for the resulting code after applying Gray map G is 0.400 for a code of length $5 \cdot 3 = 15$. If we increase the degree of D to four, the resulting code has 3^8 codewords and information rate 0.533, but the minimum homogeneous distance drops to two, so using this code one would only be able to detect one error, but not be able to correct it.

As one can see there is a trade-off between information rate and the minimum homogeneous distance of a code over $\mathbb{Z}/p^\ell\mathbb{Z}$ (and therefore the information rate and minimum Hamming distance for the respective code over $\mathbb{F}_p$). Intuitively, we can keep increasing the degree of the divisor until the size of the basis of $\mathcal{L}$ is equal to the number of points. At that point our generating matrix would be a square matrix, and when we put the matrix in standard form it has the potential (if there are enough units in the matrix) to become a diagonal matrix with 1 along the diagonal. The codewords from such a matrix would fill the entire ambient space so it would not be a proper code.

We summarize our findings in Table 4.1 and 4.2. We fixed the value of d over specific finite fields and then generated codes for all monic polynomials of degree d that created a hyperelliptic curve. We did this with low divisor degree, due to

computational limitations. After we identified which hyperelliptic curves had the highest number of affine points, we then selected those polynomials and generated codes, increasing the degree of the divisor as high as computationally feasible. Once we reached over 2,000,000 codewords, computing the minimum homogeneous distance became infeasible.

Both tables contain the highest information rate of all non-linear codes computed over $\mathbb{F}_p$ resulting from an affine linear code over $\mathbb{Z}/p^\ell\mathbb{Z}$ of a given code rank. Since the affine linear codes are free modules, they have a code rank which is the rank of the free module generated by the rows of the generating matrix. We also include the minimum Hamming distance for the corresponding code. Table 4.1 is limited to codes from curves over $\mathbf{W}_2(\mathbb{F}_p)$ and Table 4.2 is limited to codes from curves over $\mathbf{W}_3(\mathbb{F}_p)$.

We note that the information rates for codes from curves over $\mathbf{W}_3(\mathbb{F}_p)$ are much lower than those over $\mathbf{W}_2(\mathbb{F}_p)$, while the minimum Hamming distance is much higher. The Gray map G introduces significant amounts of extra information as the length of the Witt vector and characteristic of the base field increases.

We have less information for curves over $\mathbf{W}_3(\mathbb{F}_p)$ because it takes significantly longer to compute the minimum homogeneous distance, all other variables staying the same. This is due to the fact that codes over $\mathbb{Z}/p^3\mathbb{Z}$ have significantly more codewords than codes over $\mathbb{Z}/p^2\mathbb{Z}$. While the information rates are smaller, there are still more codewords in the code, but the ambient space is larger by a factor of p . Without more a more efficient algorithm for computing minimum distances over codes, we are unable to efficiently collect more information.

It is also of interest how easily one can find codes with the highest information rate provided in Table 4.1 and 4.2. As an example, we collected all codes generated from polynomials over $\mathbb{F}_7$ of degree $d = 5$ and code rank two using Witt vectors of length three. In Figure 4.1 we provide a histogram of their information rates. Note that there is more than one code that provided the best information rate, though there are a significant number of polynomials that generated information rates less than the best found rate. This shows that while ideally, one would want more codes

Table 4.1: Non-linear codes over $\mathbb{F}_p$ with highest information rate and their minimum Hamming distance generated from lifted hyperelliptic curves over $\mathbf{W}_2(\mathbb{F}_p)$.

p	$\deg(f(x_0))$	Code Rank			
		1	2	3	4
3	3	0.333, 4	0.500, 4	0.400, 4	0.533, 2
	4	0.222, 6	0.444, 2		
	6		0.333, 4	0.400, 2	
	7		0.333, 4		
5	3	0.267, 8	0.400, 4	0.130, 24	0.170, 20
	4	0.200, 8	0.267, 4	0.150, 24	0.200, 16
	5		0.200, 8	0.240, 4	0.178, 20
	6		0.200, 12	0.120, 24	0.160, 24
	8			0.200, 8	0.160, 16
7	3	0.190, 12	0.286, 6	0.071, 54	
	4	0.143, 12			
	5		0.143, 12		
	6		0.143, 12		

Table 4.2: Non-linear codes over $\mathbb{F}_p$ with highest information rate and their minimum Hamming distance generated from lifted hyperelliptic curves over $\mathbf{W}_3(\mathbb{F}_p)$.

p	$\deg(f(x_0))$	Code Rank			
		1	2	3	4
3	3	0.167, 12	0.250, 12	0.200, 12	0.267, 6
	4	0.111, 18	0.222, 6		
	6		0.167, 12	0.200, 6	
5	3	0.080, 40	0.027, 140	0.040, 120	
	4	0.060, 40	0.08, 20	0.045, 120	
	5		0.060, 40	0.040, 100	
	6		0.60, 60	0.036, 120	
7	3	0.015, 252	0.010, 420		
	4	0.031, 84			
	5		0.031, 84		
	6		0.020, 210		

with higher information rates, the rates given a particular rank and degree are not heavily stacked towards the best information rate. The choice of polynomial does matter when selecting a hyperelliptic curve and trying to generate good codes.

Figures 4.2 and 4.3 show that the distribution of these codes can vary across the same base field. These figures give an example of codes from polynomials over $\mathbb{F}_5$ of degree four and five with code rank one and two, respectively, whose histograms have very different shapes.

We provide an additional visualization of the results in Figures 4.4, 4.5, and 4.6. Here, we provide a scatter plot of the information rate against the minimum Hamming distance. The figures are split apart for different characteristics, and we provide distinct plots for codes from Witt vectors of length two and length three. We also colored the observations based on the degree d of $f(x_0)$. Every plot shows that the inverse relationship between information rate and minimum distance. Moreover, we can also see that the degree of the polynomial is evenly distributed throughout the graph. Note though that for a fixed minimum distance, polynomials with lower degrees achieved better information rates.

These results all came from hyperelliptic curves. However, this method to generate codes from curves is not limited to just hyperelliptic curves. As we saw in Section 3.4.2, Algorithm 2 can be used to generate lifts of singular curves. As such, we can create codes from these lifts. From the formula to create the generating matrix in Algorithm 4, the more affine points a curve has allows one to create a code of greater length. In some instances for a fixed characteristic and degree d , the singular curves contain more affine points than the hyperelliptic curves.

For example, when examining curves over $\mathbb{F}_3$ of degree $d = 4$, the most number of $\mathbb{F}_3$ -rational points on an affine hyperelliptic curve is four, while the most number of $\mathbb{F}_3$ -rational points on an affine singular curve is six!

In Table 4.3 we provide statistics similar to Tables 4.1 and 4.2 for non-linear codes generated from singular curves over $\mathbb{F}_3$. Note that some singular curves outperformed hyperelliptic curves when it came to the same degree and code rank. For

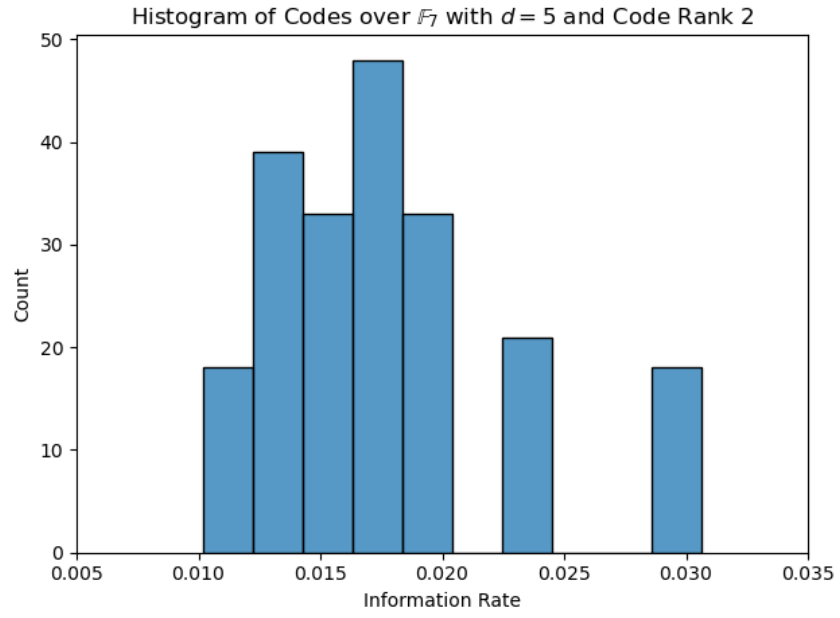


Figure 4.1: Histogram of the information rate of non-linear codes of code rank two with degree $d = 5$ over $\mathbb{F}_7$ from the ring of Witt vectors of length three.

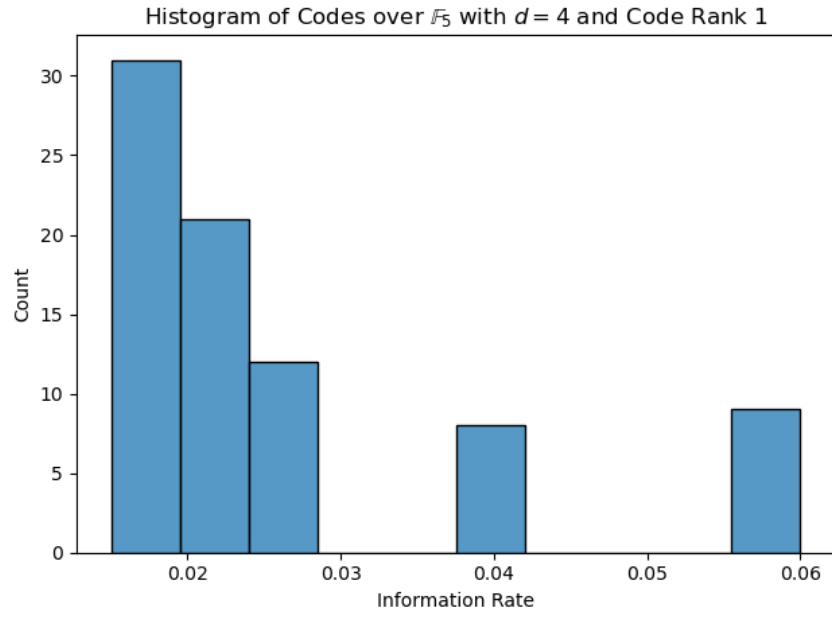


Figure 4.2: Histogram of the information rate of non-linear codes of code rank one with degree $d = 4$ over $\mathbb{F}_5$ from the ring of Witt vectors of length three.

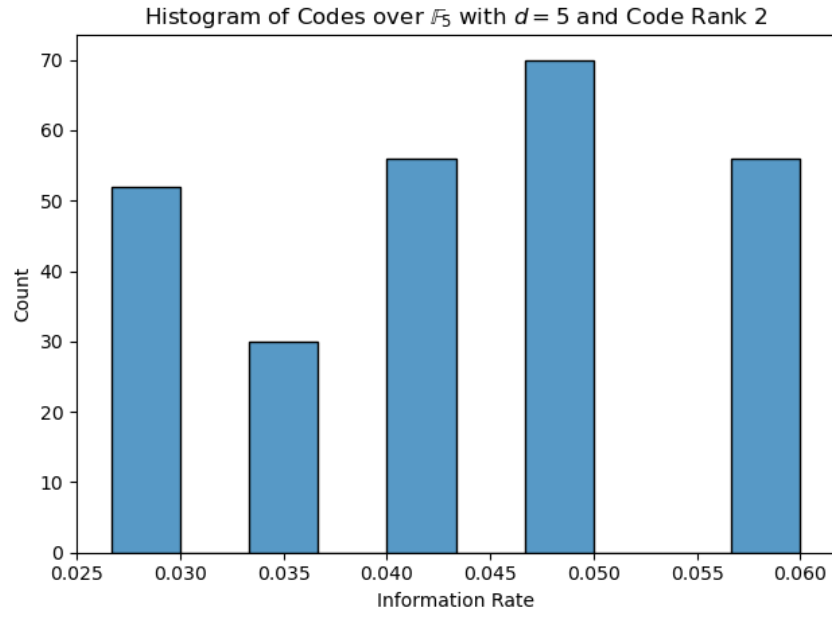


Figure 4.3: Histogram of the information rate of non-linear codes of code rank two with degree $d = 5$ over $\mathbb{F}_5$ from the ring of Witt vectors of length three.

Information Rate vs. Minimum Hamming Distance for $p = 3$

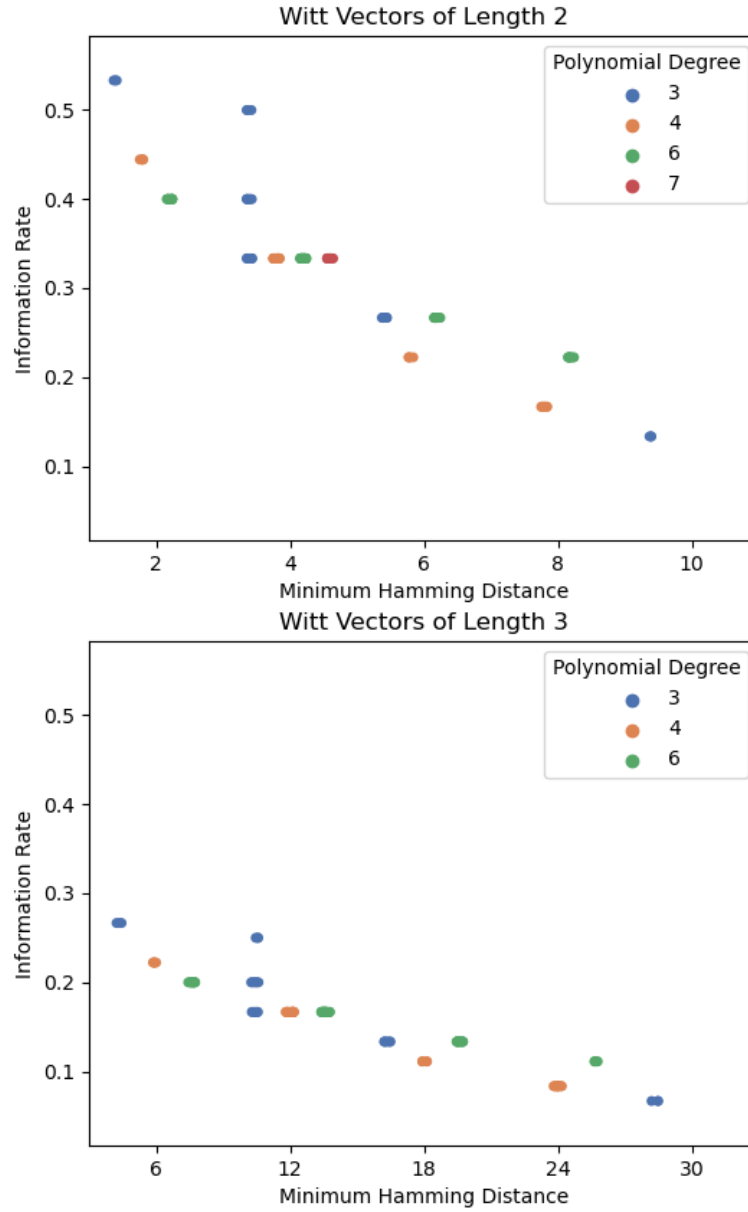


Figure 4.4: Scatter plots of information rate versus minimum Hamming distance for characteristic $p = 3$ grouped by polynomial degree d .

Information Rate vs. Minimum Hamming Distance for $p = 5$

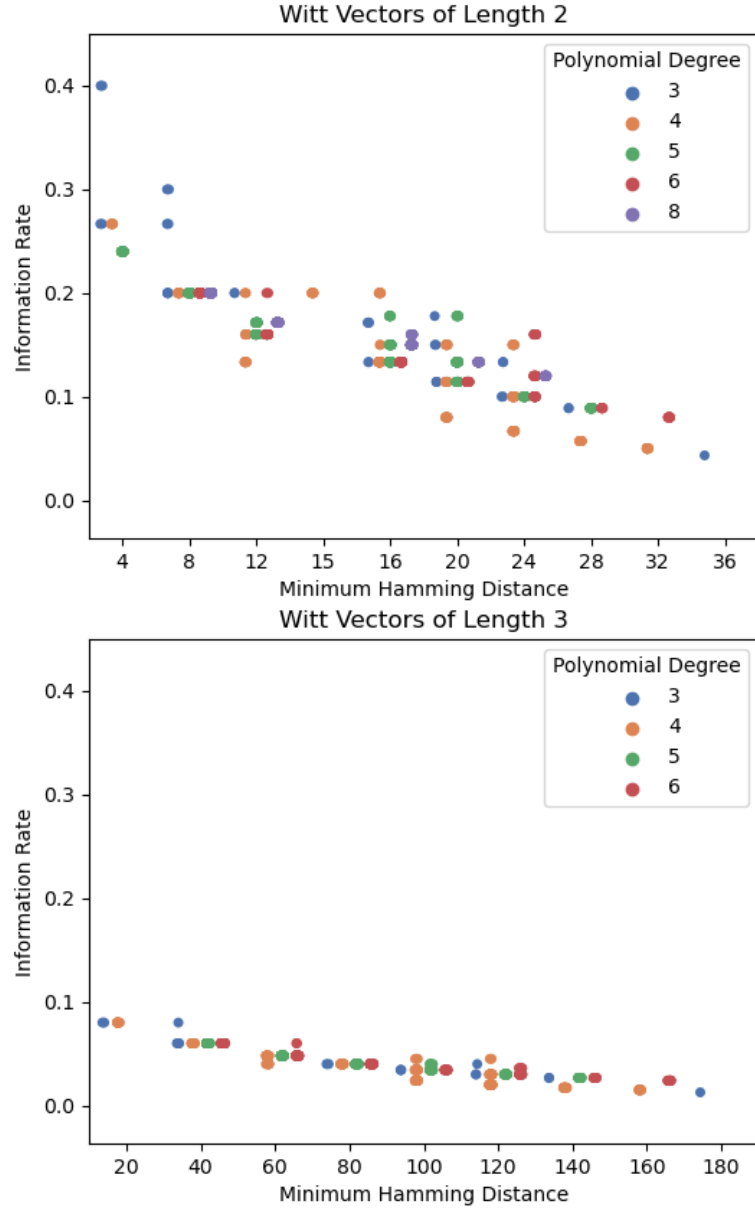


Figure 4.5: Scatter plots of information rate versus minimum Hamming distance for characteristic $p = 5$ grouped by polynomial degree d .

Information Rate vs. Minimum Hamming Distance for $p = 7$

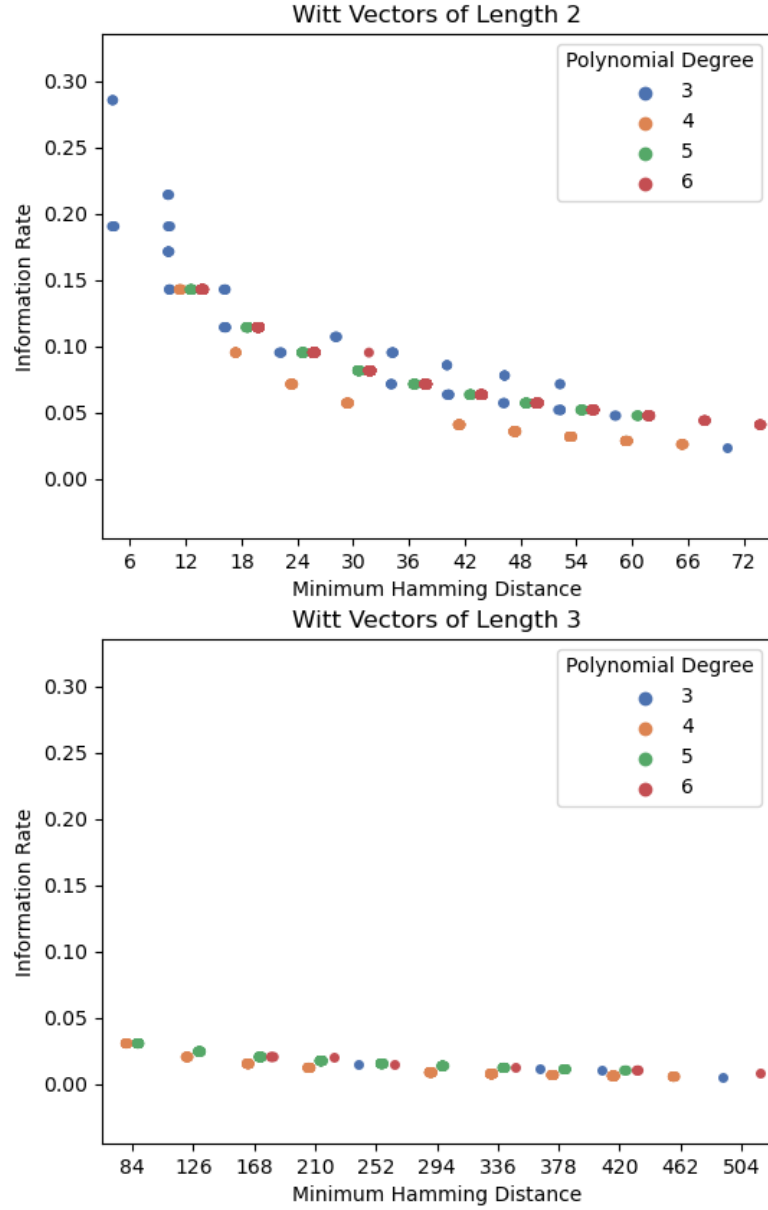


Figure 4.6: Scatter plots of information rate versus minimum Hamming distance for characteristic $p = 7$ grouped by polynomial degree d .

example, over $\mathbf{W}_2(\mathbb{F}_3)$, the best code with code rank three from an elliptic curve had information rate 0.4 with minimum Hamming distance four, while the best code with same code rank from a singular curve with the same polynomial degree had a minimum Hamming distance of four, but an information rate of 0.5. Some did perform worse; for example the best code from $\mathbf{W}_3(\mathbb{F}_3)$ and polynomial degree three and rank two had a smaller information rate in the singular case. However this shows that there is value computing these codes with both hyperelliptic and singular curves.

The benefit of selecting curves with more affine points is that it allows one to construct linear codes with greater code rank. By choosing an appropriate divisor D , one can create codes with higher code rank, and therefore increase the number of codewords.

There remains a question of the practicality of these results, and what further results could be found by extending the length of the Witt vectors. Using the results of Finotti in (7), Algorithm 2 can be expanded to lift hyperelliptic curves to curves over the ring of Witt vectors of length $\ell > 3$. One would be able to produce affine linear codes over $\mathbb{Z}/p^\ell\mathbb{Z}$ and apply the Gray map. However, the length of these codes increase exponentially as ℓ increases. This will result in codes with much higher minimum Hamming distances, but they will require significantly more computer storage to be implemented.

While there has been some research interest in binary non-linear codes, there has not been as much interest in non-linear p -ary codes for $p > 2$. These methods provide a way to generate non-linear p -ary codes with high minimum Hamming distance at the cost of low information rate. Further advances in algebraic coding theory may result in techniques to generate better p -ary codes.

To finish, we compare our results to best known *linear* codes over $\mathbb{F}_p$. All comparisons were made using the website

codetables.de,

Table 4.3: Non-linear codes over $\mathbb{F}_3$ with highest information rate and their minimum Hamming distance generated from lifted singular curves over $\mathbf{W}_\ell(\mathbb{F}_3)$.

ℓ	$\deg(f(x_0))$	Code Rank		
		1	2	3
2	3	0.333, 4	0.333, 4	0.500, 4
	4	0.222, 6	0.444, 4	0.333, 8
	6		0.333, 4	0.400, 2
3	3	0.167, 12	0.167, 12	0.250, 12
	4	0.111, 18	0.222, 12	0.167, 24
	6		0.167, 12	0.200, 6

which is maintained by Markus Grassl. To compare non-linear codes to linear codes, we compared based on the code length and the number of codewords. Note that for a $[n, k, d]$ linear code over $\mathbb{F}_p$, it has exactly p^k many codewords.

We found multiple non-linear codes that matched the best minimum Hamming distance for linear codes. However, for all these curves we had that $\mathcal{L} = \{1\}$. Moreover, this only occurred for curves with genus $g = 1$, i.e., elliptic curves. Thus the generating matrix over $\mathbb{Z}/p^\ell\mathbb{Z}$ for these codes was just a row vector of ones.

We then compared codes that had the same parameters for code length and number of codewords and had a minimum Hamming distance close to the best known minimum Hamming distance for a linear code of that same length and size. While the hyperelliptic curves were off by two or more, there were several singular codes that only off by one. For example, consider the singular curve given by

$$E/\mathbb{F}_3: \quad y_0^2 = x_0^4 + 2x_0^2 + 1,$$

which has lift $\mathcal{G}(\mathbf{E})$

$$(y_0, G_1)^2 = (1, 0)(x_0, F_1)^4 + (2, 1)(x_0, F_1)^2 + (1, 0),$$

with

$$F_1 = x_0^5 + 2x_0,$$

$$G_1 = 2x_0^6 + x_0^4 + 2x_0^2.$$

Choosing the divisor $D = 4(\mathcal{O}_1 + \mathcal{O}_2)$, we have the basis for $\mathcal{L}$ is $\{1, x_0, x_0^2\}$, resulting in generator matrix (over $\mathbb{Z}/3^2\mathbb{Z}$)

$$\begin{bmatrix} 1 & 0 & 0 & 2 & 8 & 1 \\ 0 & 1 & 0 & 1 & 8 & 8 \\ 0 & 0 & 1 & 7 & 3 & 1 \end{bmatrix}.$$

The code from this generating matrix has length 18 with 3^6 codewords and minimum Hamming distance eight. The best known minimum Hamming distance for a linear code with the same length and number of codewords is 9.

From this example we see that there are non-linear codes generated using the method with minimum Hamming distance and information rate close to the same parameters for linear codes with the same length and number of codewords. Unfortunately, there do not currently exist any websites with tables storing best known minimum Hamming distance for non-linear p -ary codes with $p > 2$. Despite this, we can see from these results that there lies more potential in exploring these areas of algebraic coding theory and seeing if one can generate good non-linear codes.

Bibliography

- [1] Claude Carlet, *$\mathbb{Z}_{2^k}$ -linear codes*, IEEE Transactions on Information Theory **44** (1998), no. 4, 1543–1547.
- [2] Jonathan Chan, Soonho Kwon, and Michael Seaman, *On the distribution of discriminants over a finite field*, arXiv preprint arXiv:1812.06231 (2018).
- [3] Ioana Constantinescu and Werner Heise, *A metric for codes over residue class rings*, Problemy Peredachi Informatsii **33** (1997), no. 3, 22–28.
- [4] Max Deuring, *Die typen der multiplikatorenringe elliptischer funktionenkörper: G. herglotz zum 60. geburtstag gewidmet*, Abhandlungen aus dem mathematischen Seminar der Universität Hamburg, vol. 14, Springer, 1941, pp. 197–272.
- [5] Ilya Dumer, Daniele Micciancio, and Madhu Sudan, *Hardness of approximating the minimum distance of a linear code*, IEEE Transactions on Information Theory **49** (2003), no. 1, 22–37.
- [6] Luís R. A. Finotti, *Degrees of the elliptic teichmüller lift*, Journal of Number Theory **95** (2002), no. 2, 123–141.
- [7] ———, *Minimal degree liftings of hyperelliptic curves*, Journal of Mathematical Sciences-University of Tokyo **11** (2004), no. 1, 1–48.
- [8] ———, *Minimal degree liftings in characteristic 2*, Journal of pure and applied Algebra **207** (2006), no. 3, 631–673.

- [9] ———, *Computations with witt vectors of length 3*, Journal de théorie des nombres de Bordeaux **23** (2011), no. 2, 417–454.
- [10] ———, *Computations with witt vectors and the greenberg transform*, International Journal of Number Theory **10** (2014), no. 06, 1431–1458.
- [11] Fernando Q Gouvêa, *p-adic numbers*, Springer, 1997.
- [12] Ziling Heng and Qin Yue, *Generalized gray map and a class of p-ary nonlinear codes*, Finite Fields and Their Applications **36** (2015), 81–97.
- [13] Thomas Honold, *Characterization of finite frobenius rings*, Archiv der Mathematik **76** (2001), no. 6, 406–415.
- [14] Nathan Jacobson, *Basic algebra II*, 2009.
- [15] Jonathan Lubin, Jean-Pierre Serre, and John Tate, *Elliptic curves and formal groups*, Lecture notes prepared in connection with the seminars held at the Summer Institute on Algebraic Geometry, Whitney Estate, Woods Hole, Massachusetts, vol. 6, 1964.
- [16] Alfred Menezes, Robert Zuccherato, and Yi-Hong Wu, *An elementary introduction to hyperelliptic curves*, Faculty of Mathematics, University of Waterloo, 1996.
- [17] Bjorn Poonen, *Computing torsion points on curves*, (2001).
- [18] Takakazu Satoh, *The canonical lift of an ordinary elliptic curve over a finite field and its point counting*, J. Ramanujan Math. Soc. **15** (2000), no. 4, 247–270.
- [19] Joseph H Silverman, *The arithmetic of elliptic curves*, vol. 106, Springer, 2009.
- [20] Jacobus Hendricus Van Lint, *Introduction to coding theory*, vol. 86, Springer Science & Business Media, 1998.

- [21] Alexander Vardy, *The intractability of computing the minimum distance of a code*, IEEE Transactions on Information Theory **43** (1997), no. 6, 1757–1766.
- [22] José Voloch, *Torsion points on $y^2 = x^6 + 1$* , arXiv preprint math/9710224 (1997).
- [23] José Voloch and Judy Walker, *Euclidean weights of codes from elliptic curves over rings*, Transactions of the American Mathematical Society **352** (2000), no. 11, 5063–5076.
- [24] ———, *Homogeneous weights and exponential sums*, Finite Fields and Their Applications **9** (2003), no. 3, 310–321.
- [25] Judy Walker, *The Nordstrom-Robinson code is algebraic-geometric*, IEEE Transactions on Information Theory **43** (1997), no. 5, 1588–1593.
- [26] ———, *Algebraic geometric codes over rings*, Journal of pure and applied Algebra **144** (1999), no. 1, 91–110.
- [27] Bahattin Yildiz and Zeynep Ödemiş Özger, *Generalization of the Lee weight to $\mathbb{Z}_{p^k}$* , TWMS Journal of Applied and Engineering Mathematics **2** (2012), no. 2, 145–153.

Vita

Originally born in Ohio, Justin Groves moved at a young age and grew up in Iva, South Carolina. After high school, he attended Winthrop University and received a Bachelor of Science degree in Mathematics. Before graduating with his undergraduate degree, he knew he wanted to attend graduate school. He chose to attend the University of Tennessee, Knoxville to pursue a Doctor of Philosophy degree in Mathematics with a concentration in Algebra. His research interest includes algebraic geometry and elliptic curves. After graduation he plans to pursue a career in data science. He is incredibly grateful for all the support from his wife, family, and friends throughout his career.