R. Lemaire

Enumerating Superelliptic Function Fields over Finite Fields

Bachelor thesis

26 June 2026

Thesis supervisors: dr. T.C. Streng
dr. N. Zidarič

Leiden University
Mathematical Institute
Leiden Institute of Advanced Computer Science

Abstract

We present an algorithm that enumerates the isomorphism classes of superelliptic function fields of exponent m over a finite field $\mathbb{F}_q$ such that m divides $q-1$, given some restrictions on the square-free factorisations of the related binary forms. The algorithm runs in quasilinear time in the size of its output, and its memory complexity is logarithmic in q. Our work is a generalisation of the algorithms by Everett Howe for enumerating hyperelliptic function fields [How25a; How25b].

Contents

Nomenclature

$B_m(K)$	The class of binary forms that define superelliptic function fields of exponent m over the field K	Definition 3.19
$B_m^D(K)$	The class of binary forms in $\mathcal{B}_m(K)$ of square-free factorisation type equivalent to D	Definition 3.32
$\mathbb{F}_q$	A finite field of q elements	
$\mathrm{Gal}\,(F/E)$	The Galois group of the Galois extension F/E	
$\widetilde{K}$	The field of constants of an algebraic function field F/K	Definition 2.2
$O(g)$	The class of functions that grow at most as fast as the function g asymptotically	Section 2.4
$\widetilde{O}(g)$	The class of functions that grow at most as fast as $g \cdot \log(g)^k$ asymptotically for some integer $k > 0$	Section 2.4
$\Theta(g)$	The class of functions that grow equally as fast as the function g asymptotically	Section 2.4
$\mathbb{P}^1(K)$	The projective line over the field K	Definition 3.24
$\mathrm{PGL}_2\,(K)$	The projective linear group over the field K	Definition 2.4
$P_m(K)$	The class of polynomials that define superelliptic function fields of exponent m over the field K	Definition 3.13
$R_m(K)$	The class of rational functions that define superelliptic function fields of exponent m over the field K	Definition 3.5
$S_m(K)$	The class of superelliptic function fields of exponent m over the field K	Definition 3.1
$\mathbb{Z}/m\mathbb{Z}$	The integers modulo m	
$\mathrm{zeroes}(G)$	The set of distinct zeroes in $\mathbb{P}^1(K)$ of the binary form G	Definition 3.25

1 Introduction

The goal of this thesis is to give an algorithm that enumerates isomorphism classes of superelliptic function fields over a given finite field $\mathbb{F}_q$. We only consider superelliptic function fields of exponent m such that $m \mid q-1$, which are cyclic Kummer extensions $\mathbb{F}_q(x,y)$ of the rational function field $\mathbb{F}_q(x)$ defined by an equation of the form

$$y^m = k \cdot \prod_{i=1}^{r} g_i(x)^{n_i}, \tag{1}$$

for some integer $m \geq 2$ that divides $q-1$, $k \in \mathbb{F}_q^*$, positive $r \in \mathbb{Z}$, non-zero $n_i \in \mathbb{Z}$ satisfying $\gcd(m, n_1, \ldots, n_r) = 1$, and distinct monic irreducible polynomials $g_i \in \mathbb{F}_q[x]$.

We show that superelliptic function fields can equivalently be described by equations of the form $y^m = G(X,Z)$, where $G \in \mathbb{F}_q[X,Z]$ is a homogeneous bivariate polynomial, also known as a *binary form*, satisfying similar constraints given in Definition 3.19. These binary forms, up to scaling by m^{th} powers in $\mathbb{F}_q$, permit an action of the group $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$, and we show that their orbits under the action of this group are in bijection with the isomorphism classes of superelliptic function fields in Theorem 3.21.

We present an algorithm to enumerate the orbits of these binary forms in Section 4.2. In addition to the finite field $\mathbb{F}_q$ and the exponent m, our algorithm takes a *square-free factorisation type* as input, which specifies the degrees of the square-free binary forms in the square-free factorisations of the binary forms that we are enumerating. Concretely, our algorithm gives exactly one representative of every orbit that contains a binary form of the specified square-free factorisation type.

Furthermore, we analyse the complexity of our algorithm as a function of q, and we show that it takes quasilinear time in its output size, which means the running time increases linearly in the number of orbits we enumerate, barring factors logarithmic in q. We also show that the algorithm uses only $O(\log q)$ memory, and we provide an implementation in the programming language Julia [Bez+17] using the computer algebra system OSCAR [OSC26].

We will start by discussing the necessary background knowledge on algebraic function fields, Kummer theory, Julia, OSCAR, and algorithmic complexity in Section 2, which some readers may already be familiar with. In Section 3 we define superelliptic function fields and their isomorphisms using an approach that is specific to this thesis and particularly practical for our algorithmic purposes. Our algorithm is based on previous work by Everett Howe on hyperelliptic function fields [How25b], and we discuss both his algorithms and our own in Section 4.

2 Background

In Sections 2.1 and 2.2, we will discuss the preliminary knowledge necessary for understanding superelliptic function fields and their isomorphisms as described in Section 3. We assume undergraduate algebra knowledge on groups, rings, fields, and Galois theory. We will also discuss the language features of Julia and OSCAR, and present some basic concepts from algorithmic complexity analysis in Sections 2.3 and 2.4.

We will start by studying general algebraic function fields in Section 2.1, with a focus on the rational function field in Section 2.1.1. Additionally, we take a look at cyclic Kummer extensions in Section 2.2, because we will see in Section 3 that, in our case, superelliptic function fields are cyclic Kummer extensions of the rational function field.

2.1 Algebraic Function Fields

Let us start by studying algebraic function fields. Throughout this section, we let K be an arbitrary field, and we will consider algebraic function fields as field extensions of K.

Definition 2.1 (cf. [Sti08, Def 1.1.1])**.** An *(algebraic) function field (of one variable over K)* is a field extension F/K such that there exists an $x \in F$ for which $F/K(x)$ is finite and $K(x)/K$ is transcendental.

Note that after making a choice of a transcendental $x \in F$, we can see such a function field as an algebraic extension of a rational function field $K(x)$. We will discuss this choice, and the special case $F = K(x)$, in Section 2.1.1.

In Section 3, and in particular Definition 3.1, we will concern ourselves with function fields that do not contain elements that are algebraic over K, apart from the elements of K itself.

Definition 2.2. The *field of constants* of F/K, denoted by $\widetilde{K}$, consists of the $z \in F$ that are algebraic over K. We call K *algebraically closed in* F if $\widetilde{K} = K$.

2.1.1 Rational Function Field

Now that we have a general background on algebraic function fields, we will study the rational function field $K(x)/K$ in more detail. Here x is transcendental over K, so $K(x)$ is isomorphic to the field of rational functions over K. We will start by investigating the generators of $K(x)/K$.

Theorem 2.3. *Suppose that x is transcendental over K. The generators of $K(x)$ over K are the elements $\frac{ax+b}{cx+d}$, where $a, b, c, d \in K$ satisfy $ad - bc \neq 0$.*

Proof. Let $a, b, c, d \in K$ be such that $ad - bc \neq 0$. First of all, note that such c and d cannot both be zero, so $t := \frac{ax+b}{cx+d}$ is well-defined. Additionally, we may verify $x = \frac{dt-b}{-ct+a} \in K(t)$, so $K(t) = K(x)$.

Conversely, suppose that $t \in K(x)$ is such that $K(t) = K(x)$. We can write $t = \frac{f(x)}{g(x)}$ for some relatively prime non-zero polynomials $f, g \in K[X]$. Now $h(X) := f(X) - tg(X) \in K[t][X]$ satisfies $h(x) = 0$. We will show that h is, up to a unit of $K(t)$, the minimal polynomial of x over $K(t)$.

Suppose that h is reducible in $K(t)[X]$, i.e., there exist $h_1, h_2 \in K(t)[X]$ of positive degree with respect to X such that $h = h_1 h_2$. As a result of Gauss's Lemma [Lan12, Corollary IV.2.2], we may assume $h_1, h_2 \in K[t][X]$. Since h is linear in t, at least one of h_1 and h_2 is constant with respect to t. Without loss of generality, assume $h_1 \in K[X]$. Then since h_1 divides $h(X) = f(X) - tg(X)$, it divides both f and g in $K[t][X]$. However, we know that h_1 has positive degree with respect to X and f and g are relatively prime, so this is a contradiction. We conclude that h is irreducible over $K(t)[X]$, and thus is a scalar multiple of the minimal polynomial of x over $K(t)$.

As we have $K(t) = K(x)$, we know $\deg_X h = [K(t,x) : K(t)] = 1$. By definition of h we know $\max\{\deg f, \deg g\} = \deg_X h = 1$, so $f(x) = ax + b$ and $g(x) = cx + d$ for some $a, b, c, d \in K$. Since f and g are not multiples of each other, we conclude $ad - bc \neq 0$. □

Due to the restriction $ad - bc \neq 0$ the matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{Mat}_2(K)$ will be invertible. Clearly the corresponding generator $\frac{ax+b}{cx+d}$ is equal to that of another invertible matrix if and only if these matrices are scalar multiples of one another. This motivates an action of the *projective linear group* $\mathrm{PGL}_2(K) := {}^{\mathrm{GL}_2(K)}/_{K^*}$ on the rational functions.

Definition 2.4. The group $\mathrm{PGL}_2(K)$ acts on the rational functions $f \in K(x)$ from the left as follows: let $\Gamma \in \mathrm{PGL}_2(K)$ be represented by $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{GL}_2(K)$ and let $h_\Gamma(x) := \frac{ax+b}{cx+d}$. Then $\Gamma(f) := f \circ h_{\Gamma^{-1}} = f\left(\frac{dx-b}{-ax+c}\right)$.

The group action axioms follow from the observation that for any $\Gamma_1, \Gamma_2 \in \mathrm{PGL}_2(K)$ we have $h_{\Gamma_1} \circ h_{\Gamma_2} = h_{\Gamma_1 \Gamma_2}$.

2.2 Kummer Theory

Kummer theory is the study of algebraic field extensions that adjoin n^{th} roots of elements to some field E. Central in this theory are the n^{th} roots of unity, that is, the $\zeta \in \overline{E}$ such that $\zeta^n = 1$. We call ζ a *primitive n^{th} root of unity* if the order of ζ in $\overline{E}^*$ is exactly n [Ste25, Section 24, Cyclotomic Extensions].

Definition 2.5 (cf. [Rom05, Section 15.1])**.** A Galois extension F/E is *Kummer of exponent n* if E contains a primitive n^{th} root of unity, and $\mathrm{Gal}(F/E)$ is an Abelian group of exponent n.

For the remainder of this section we will restrict ourselves to Kummer extensions with cyclic Galois group, in which case the exponent will simply be the degree of the extension. Additionally, we will denote by ζ a fixed primitive n^{th} root of unity contained in our base field. The following is the fundamental theorem of Kummer theory for cyclic Kummer extensions.

Theorem 2.6 (cf. [Lan12, Thm. IV.6.2])**.** *Let E be a field containing a primitive n^{th} root of unity ζ. Then the following hold:*

1. *Every cyclic Kummer extension F/E of degree n is of the form $F = E(\alpha)$, for some $\alpha \in F$ such that $\alpha^n \in E$;*
2. *Let $F = E(\alpha)$ for some $\alpha \in \overline{E}$ such that $\alpha^n \in E$. Then F/E is cyclic Galois of degree d, where d is the smallest positive divisor of n such that $\alpha^d \in E$.*

Proof. Since this is a rather well-known theorem, we will only give a sketch of the proof, and refer to Lang [Lan12, Thm. IV.6.2] or Stevenhagen [Ste25, Thm. 25.16] for the full proof.

For the first statement, we shall take any $\sigma \in \mathrm{Gal}\,(F/E)$ that generates $\mathrm{Gal}\,(F/E)$, and choose an $\alpha \in F$ such that $\sigma(\alpha) = \zeta\alpha$. Choosing such an α can be done using Hilbert's Theorem 90 as in [Lan12] or the Lagrange resultant as in [Ste25]. Then we make the observations that α has n distinct conjugates under $\mathrm{Gal}\,(F/E)$, and that α^n is fixed by $\mathrm{Gal}\,(F/E)$.

The second statement relies on the observation that F/E is Galois and that for any $\sigma \in \mathrm{Gal}\,(F/E)$ we have $(\sigma(\alpha)/\alpha)^n = 1$, which gives us an injection $\mathrm{Gal}\,(F/E) \hookrightarrow \langle\zeta\rangle$. This means $\mathrm{Gal}\,(F/E)$ is cyclic of some order $d \mid n$, and $\sigma(\alpha^d) = \sigma(\alpha)^d = \alpha^d$, so $\alpha^d \in K$. □

Corollary 2.7. *Suppose that F/E is a cyclic Kummer extension of degree n and write $F = E(\alpha)$ with $\alpha^n \in E$. Then $\mathrm{Gal}\,(F/E) = \langle\sigma : \alpha \mapsto \zeta\alpha\rangle$.*

Proof. We have $\alpha^n \in E$, so α is a root of $X^n - \alpha^n \in E[X]$. Additionally $[E(\alpha) : E] = [F : E] = n$ and $X^n - \alpha^n$ is a monic polynomial of degree n, so it is the minimal polynomial of α over E. As ζ is a primitive n^{th} root of unity, the n distinct roots of f_E^α are given by $\{\zeta^i\alpha : 0 \leq i < n\}$. Therefore, the map $\sigma : \alpha \mapsto \zeta\alpha$ is a bijection of order n of the roots of f_E^α. Since $\mathrm{Gal}\,(F/E) \cong C_n$, it is generated by any such element of order n. □

2.2.1 Generators of Cyclic Kummer Extensions

In Theorem 2.6 we saw that any cyclic Kummer extension of degree n permits a generator whose n^{th} power lies in the base field. Naturally, one might wonder to what extent this generator is unique.

Lemma 2.8. *Suppose that F/E is a cyclic Kummer extension of degree $n > 1$, and let $\alpha, \beta \in F$ be such that $F = E(\alpha) = E(\beta)$ and $\alpha^n, \beta^n \in E$. Then $\beta = k\alpha^m$ for some $k \in E^*$ and $m \in \mathbb{Z}$ coprime to n.*

Proof. By considering $\{\alpha^i : 0 \leq i < n\}$ as an E-basis for $F = E(\alpha)$, we can write $\beta = f(\alpha)$ for some non-zero polynomial $f(X) = \sum_{i=0}^{m-1} a_i X^i \in E[X]$. Due to Corollary 2.7 we know that the Galois group is given by both $\langle\sigma : \alpha \to \zeta\alpha\rangle$ and $\langle\tau : \beta \to \zeta\beta\rangle$. Therefore, there exists an $m \in \mathbb{Z}$ such that τ^m is σ, and thus τ^m has order n in $\mathrm{Gal}\,(F/E)$. Then m is coprime to n. Now we consider the following identity:

$$f(\zeta\alpha) = f(\sigma(\alpha)) = \sigma(f(\alpha)) = \sigma(\beta) = \tau^m(\beta) = \zeta^m\beta = \zeta^m f(\alpha).$$

By expanding f we find $\sum_{i=0}^{m-1} a_i\zeta^i\alpha^i = \sum_{i=0}^{m-1} a_i\zeta^m\alpha^i$. Since the representation of elements of F with respect to our E-basis is unique, we find that for $0 \leq i < n$ we have $a_i\zeta^i = a_i\zeta^m$. As ζ is a primitive n^{th} root of unity, we get $a_i = 0$ or $i = m$, so we have $f(X) = a_m X^m$ with $a_m \in E^*$ because $\beta \neq 0$. □

Theorem 2.9. *Suppose that F/E is a cyclic Kummer extension of degree $n > 1$, and let $\alpha \in F$ be such that $F = K(\alpha)$ and $\alpha^n \in E$. Then given $\beta \in F$, we have $F = E(\beta)$ and $\beta^n \in E$ if and only if $\beta = k\alpha^m$ for some $k \in E^*$ and $m \in \mathbb{Z}$ coprime to n.*

Proof. Lemma 2.8 already proves that any such β be of the form $k\alpha^m$. Therefore it suffices to show that any $\gamma = k\alpha^m$ also satisfies $\gamma^n \in E$ and $F = E(\gamma)$. First of all, we have that $\gamma^n = k^n(\alpha^n)^m \in E$. In order to prove $F = E(\gamma)$, we observe that there exist $r, s \in \mathbb{Z}$ such that $rm + sn = 1$. Now we can observe

$$\alpha = \alpha^{rm}\alpha^{sn} = \left(\frac{\gamma}{k}\right)^r \cdot (\alpha^n)^s \in E(\gamma).$$

Since we know $\gamma \in E(\alpha)$, we can conclude $F = E(\alpha) = E(\gamma)$. □

2.3 Computer Algebra Systems

Our algorithm for superelliptic function fields is an extension of Everett Howe's algorithm for hyperelliptic function fields [How25b]. Howe implemented his algorithm in the computer algebra system and programming language Magma [BCP97]. However, since Magma is proprietary and requires a license, we decided to implement our algorithm in Julia, an open source alternative.

The open source programming language Julia is designed from the ground up to be intuitive and efficient for problems in numerical computation and mathematics in general [Bez+17]. This instinctiveness is achieved by having built-in support for common mathematical structures and syntax, such as matrices and vectors, and by making typing optional.

When Julia packages are loaded, they are precompiled to abstract syntax trees and Julia performs type inference. Any remaining type ambiguities are solved during runtime, before finishing the compilation to machine code using a just-in-time compiler by LLVM [Juld]. Just-in-time compilation introduces some overhead at runtime, but also allows for runtime optimisations that would not be possible with static compilation [LA04, Sec. 3.1]. In practice, Julia provides performance similar to C, Rust, and Fortran on a diverse suite of micro-benchmarks [Jula].

We will make use of the computer algebra system OSCAR 1.7.3 [OSC26], which is implemented in Julia, for doing computations in polynomial rings over finite fields. Julia allows for efficient calls to libraries written in several other languages, such as Fortran and C. OSCAR makes extensive use of this feature, for example by using the FLINT C library [FLI26] for finite fields and polynomials provided by the Julia package Nemo [Fie+17].

OSCAR also benefits from Julia's flexible parametric typing system, which allows for intuitive modelling of complex mathematical structures, such as matrix spaces over polynomial rings over number fields. Furthermore, OSCAR has support for working with algebraic function fields [OSC].

Julia provides extensive support for *lazy iteration* [Julc], a programming technique where a collection of elements can yield its elements one by one without storing the entire collection simultaneously, which will be critical for our memory complexity, as we will discuss in Section 2.4. OSCAR also provides such lazy iterators for many of its collections, such as finite fields.

There are multiple tools available to analyse Julia code. The BenchmarkTools package [Julb] makes it possible to create reproducible benchmarking setups, with control over parameters such as the functions and inputs to test, the number of samples to take per function, and the ability to trigger Julia's garbage collector in between samples.

The Profiler package [Jule] provides a statistical profiler to identify the most computationally expensive parts of your code. A statistical profiler, also called a sampling profiler, takes backtraces at a regular interval. The default interval for the Profile package is 1 ms. The backtraces contain the file and line number of the code that was running at that time, as well as all function calls made to arrive at that point of execution. The number of times a function appears in these backtraces is a good indicator for the time spent in that function, although it is subject to statistical noise.

2.4 Algorithmic Complexity

The *time complexity* of an algorithm describes the asymptotic behaviour of the number of steps the algorithm needs to complete as a function of the size of its inputs. Similarly, the *memory complexity* of an algorithm describes the asymptotic behaviour of the amount of memory needed by the algorithm as a function of the input size.

A common way to quantify asymptotic behaviour of functions is *Big O notation* [Knu76]. Given two functions $f, g : \mathbb{Z}_{>0} \to \mathbb{R}_{>0}$, we write $f \in O(g)$ if there exist constants $C \in \mathbb{R}_{>0}$ and $n_0 \in \mathbb{Z}_{>0}$ such that $f(n) \leq C \cdot g(n)$ for all $n \geq n_0$. We write $f \in \Theta(g)$ if $f \in O(g)$ and $g \in O(f)$. These definitions may also be generalised to functions of multiple variables. Additionally, we will use a common extension of this notation, the *Soft O notation*, where we say 'f is quasilinear in g' and write $f \in \widetilde{O}(g)$ if $f \in O(g \cdot \log(g)^k)$ for some constant $k \in \mathbb{Z}_{>0}$.

As we will discuss in more detail in Section 4, we will be designing algorithms that have time complexity quasilinear in the size of their output. Assuming that we need at lest one operation per element of the output, this means that the algorithm's runtime is optimal up to constant and logarithmic factors. If we would accumulate all results, we would have to store the entire output simultaneously, and our memory complexity would be lower bounded by the size of the output as well. The lazy iterators of Julia and OSCAR, as described in Section 2.3, allow us to return our results one at a time instead, which allows us to achieve much lower memory complexity.

3 Superelliptic Function Fields

In Section 3, we explore the theoretical properties of superelliptic function fields. In particular, we will discuss those function fields that are cyclic Kummer extensions of the field of rational functions over a field K, and we describe their isomorphism classes.

Definition 3.1. A *superelliptic function field* of exponent $m > 1$ over K is a pair of fields (E, F) such that $K \subset E \subset F$ satisfying the following properties:

1. F/E is of degree m and $F = E(\alpha)$ for some $\alpha \in F$ such that $\alpha^m \in E$;
2. $E = K(x)$ for some $x \in E$ that is transcendental over K;
3. K is the full constant field of F/K.

We denote the class of superelliptic function fields of exponent m over K by $\mathcal{S}_m(K)$.

In the remainder of this section, we assume there exists a primitive m^{th} root of unity $\zeta \in K$, in which case the first property of Definition 3.1 is equivalent to F/E being cyclic Kummer of degree m due to Theorem 2.6. If K is a finite field, we can easily tell when it contains such a root of unity.

Proposition 3.2. *Let K be a finite field of q elements. Then K contains a primitive m^{th} root of unity if and only if m divides $q - 1$.*

Proof. Any finite subgroup of the unit group of an integral domain is cyclic [Ste22, Cor. 12.4], so there exists a $\gamma \in K^*$ such that $K^* = \langle \gamma \rangle$. If we have $m \mid q - 1$, then $\gamma^{\frac{q-1}{m}}$ is a primitive m^{th} root of unity. Conversely, if K contains a primitive n^{th} root of unity, then its order divides the group order, and we obtain $m \mid q - 1$. □

We know that F/E is cyclic Kummer of degree m, and Theorem 2.9 tells us there are many choices for a generator $\alpha \in F$ such that $\alpha^n \in E$. Similarly, Theorem 2.3 gives us all the choices for a generator x of E/K. Let us formalise making these choices.

Definition 3.3. A *model* of a superelliptic function field (E, F) of exponent m is a pair (x, y) of elements of F such that $E = K(x)$, $F = E(y)$, and $y^m \in E$.

3.1 Isomorphism Classes

In this section, we will study the isomorphisms of superelliptic function fields over any field K in terms of their models.

Definition 3.4. An *isomorphism of superelliptic function fields* (E_1, F_1) and (E_2, F_2) of exponent m over K is a ring isomorphism $\phi : F_1 \xrightarrow{\sim} F_2$ such that $\phi\big|_{E_1}$ is a ring isomorphism from E_1 to E_2 and $\phi\big|_K$ is the identity on K. If such an isomorphism exists, we say these function fields are *isomorphic* and write $(E_1, F_1) \cong (E_2, F_2)$.

As one might expect of isomorphisms, the identity map, inverses of isomorphisms, and composites of isomorphisms all satisfy these conditions. This implies that isomorphisms divide the superelliptic function fields of exponent m into isomorphism classes. We will study the isomorphisms of superelliptic function fields through the rational functions $g \in K(X)$ such that $g(x) = y^m \in E$ for some model (x, y) of the superelliptic function field (E, F). As we will see in Proposition 3.12, these are precisely the following rational functions.

Definition 3.5. Let $\mathcal{R}_m(K)$ be the set of non-constant rational functions $g \in K(X)$ of the form $g(X) = k \cdot \prod_{i=1}^{r} g_i(X)^{n_i}$ for some $k \in K^*$, positive $r \in \mathbb{Z}$, non-zero $n_i \in \mathbb{Z}$, and distinct monic irreducible polynomials $g_i \in K[X]$ satisfying $\gcd(m, n_1, \ldots, n_r) = 1$.

The main result of this section, Theorem 3.7, will describe the isomorphism classes of $\mathcal{S}_m(K)$ as the orbits of $\mathcal{R}_m(K)/(K(X)^*)^m$ under the action of the group $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(K)$. Let us define this action, and prove that it indeed satisfies the axioms for group actions.

Definition 3.6. The group $(K(X)^*)^m$ acts on $\mathcal{R}_m(K)$ through multiplication in $K(X)$. The group $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(K)$ acts from the left on the set $\mathcal{R}_m(K)/(K(X)^*)^m$ in the following way: given $(\overline{e}, \Gamma) \in (\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(K)$ and $g \in \mathcal{R}_m(K)$, the orbit of g in $\mathcal{R}_m(K)/(K(X)^*)^m$ is sent to the orbit of $\Gamma(g)^e$, where $e \in \mathbb{Z}$ is a representative for $\overline{e}$.

Proof. The action of $\mathrm{PGL}_2(K)$ on $K(X)$ of Definition 2.4 distributes over multiplication, so it preserves irreducibility and the greatest common divisor property of Definition 3.5. Exponentiation by any $e \in \mathbb{Z}$ coprime to m also preserves this property, so $\Gamma(g)^e \in \mathcal{R}_m(K)$. Since we are working modulo $(K(X)^*)^m$, exponentiation by representatives of $(\mathbb{Z}/m\mathbb{Z})^*$ in $\mathcal{R}_m(K)/(K(X)^*)^m$ is independent of representative and satisfies the axioms of group actions. Finally, these actions of $\mathrm{PGL}_2(K)$ and $\mathbb{Z}/m\mathbb{Z}$ commute because we have $\Gamma(g^e) = \Gamma(g)^e$, so their product group also acts on $\mathcal{R}_m(K)/(K(X)^*)^m$. □

Theorem 3.7. *Let K be a field containing a primitive m^{th} root of unity. The isomorphism classes of $\mathcal{S}_m(K)$, the superelliptic function fields of exponent m, may be described through the bijection*

$$\left(\mathbb{Z}/m\mathbb{Z}\right)^* \times \mathrm{PGL}_2(K) \Big\backslash \left(\mathcal{R}_m(K)\Big/(K(X)^*)^m\right) \xrightarrow{\sim} \mathcal{S}_m(K)\Big/\cong$$

given by mapping the orbit of $g \in \mathcal{R}_m(K)$ to the isomorphism class of $(E_g, F_g) := \left(K(X), K(X)[Y]/(Y^m - g(X))\right)$. The inverse is given by taking a representative $(E, F) \in \mathcal{S}_m(K)$, choosing a model (x, y) for (E, F), and mapping the class of (E, F) to the orbit of the rational function $g \in K(X)$ satisfying $y^m = g(x)$.

The majority of Section 3.1 will be dedicated to proving Theorem 3.7, and we will start by showing that (E_g, F_g) is indeed a superelliptic function field of exponent m.

Lemma 3.8. *For any g in $\mathcal{R}_m(K)$ the pair of rings (E_g, F_g) defined in Theorem 3.7 is a superelliptic function field of exponent m. Let Y_g be the class of Y in F_g. Then (X, Y_g) is a model for (E_g, F_g).*

Proof. Due to Definition 3.5 there is no $d > 1$ that divides m such that $g(X)$ is a d^{th} power in $\overline{K}(X)$, so Theorem 2.6 tells us that adjoining any root of $Y^m - g(X)$ to $K(X)$ gives us a cyclic Kummer extension of degree m, and in particular that F_g is a field. Let $\widetilde{K}$ be the algebraic closure of K in F_g. Then $\widetilde{K}(X, Y_g) \cong \widetilde{K}(X)[Y]/(Y^m - g(X))$ is cyclic Kummer of degree m over $\widetilde{K}(X)$ by the same argument. Now we observe the field inclusions

$$K(X) \subset \widetilde{K}(X) \subset \widetilde{K}(X, Y_g) \subset F_g.$$

As $[F_g : K(X)] = m$ and $[\widetilde{K}(X, Y_g) : \widetilde{K}(X)] = m$, we find $[\widetilde{K}(X) : K(X)] = 1$. We conclude that $\widetilde{K}(X) = K(X)$ and thus $\widetilde{K} = K$, so (E_g, F_g) is a superelliptic function field of exponent m. □

Proposition 3.9. *The set $\mathcal{R}_m(K)$ is the set of rational functions $g \in K(X)$ for which there exists a function field $(E, F) \in \mathcal{S}_m(K)$ of exponent m with a model (x, y) such that $y^m = g(x)$.*

Proof. Due to Lemma 3.8 we know that for any $g \in \mathcal{R}_m(K)$ we have $(E_g, F_g) \in \mathcal{S}_m(K)$ with model (X, Y_g) such that $Y_g^m = g(X)$.

Given an element (E, F) of $\mathcal{S}_m(K)$ with a model (x, y), let $g \in K(X)$ be the rational function such that $g(x) = y^m$. Since $F = E(y)$ is of degree $m > 1$ over E, we have $y \neq 0$, and thus $g \neq 0$. Now we write $g(x) = k \cdot \prod_{i=1}^{r} g_i(x)^{n_i}$ for some $k \in K^*$, non-negative $r \in \mathbb{Z}$, non-zero $n_i \in \mathbb{Z}$, and distinct monic irreducible polynomials $g_i \in K[X]$. Let d be $\gcd(m, n_1, \ldots, n_r)$. We will show that $d = 1$, which implies that r is positive and g is an element of $\mathcal{R}_m(K)$.

Consider the following element of F:

$$\alpha = y^{\frac{m}{d}} \cdot \prod_{i=1}^{r} g_i(x)^{-\frac{n_i}{d}}.$$

Observe that $\alpha^d = k \in K$, so $\alpha \in \widetilde{K}$ and thus $\alpha \in K$, because K is the full constant field of F/K. Therefore g is a d^{th} power in $K(X)$, and we find $[F : E] \leq \frac{m}{d}$ by Theorem 2.6. We conclude that $d = 1$. □

Lemma 3.10. *Let $(E, F) \in \mathcal{S}_m(K)$ be a superelliptic function field and let (x, y) be a model for (E, F). Then $(E, F) \cong (E_g, F_g)$, where $g \in \mathcal{R}_m(K)$ is the rational function such that $y^m = g(x)$.*

Proof. Since we have $F = E(y)$, $[F : E] = m$, and $y^m - g(x) = 0$, we know there exists a ring isomorphism $\psi : F \xrightarrow{\sim} K(x)[Y]/(Y^m - g(x))$ that fixes E. Additionally, the map $\chi : K(x) \to K(X)$ that fixes K and maps x to X is a ring isomorphism, because x and X are both transcendental over K. As $F_g = K(X)[Y]/(Y^m - g(X))$, we conclude that $(E, F) \cong (E_g, F_g)$. □

Lemma 3.11. *Let (E_i, F_i) be a superelliptic function field of exponent m over K and choose a model (x_i, y_i) for each $i = 1, 2$. Suppose these function fields are isomorphic with isomorphism $\phi : F_1 \xrightarrow{\sim} F_2$. Then the following are true:*

1. *$\phi(x_1) = \Gamma(x_2)$ for some $\Gamma \in \mathrm{PGL}_2(K)$;*
2. *$\phi(y_1) = h(x_2)y_2^e$ for some $h(x_2) \in K(x_2)^*$ and $e \in \mathbb{Z}$ coprime to m.*

Conversely, any ring homomorphism $\phi : F_1 \to F_2$ that fixes K and satisfies these two properties is an isomorphism of superelliptic function fields.

Proof. Let $\phi : F_1 \xrightarrow{\sim} F_2$ be such an isomorphism. Observe that $\phi(x_1)$ is a generator of $E_2 = K(x_2)$ over K. Analogously, the image of y_1 is a generator of $F_2 = E_2(y_2)$ over E_2, where instead of K being fixed, we now observe that ϕ maps E_1 to E_2 and back. Theorems 2.3 and 2.9 now give us Properties 1 and 2.

Conversely, let $\phi : F_1 \to F_2$ be a ring homomorphism that fixes K and satisfies the two given properties. The image of ϕ contains generators of F_2 over K and ϕ fixes K, so ϕ is a surjective ring homomorphism. The kernel of ϕ is an ideal of the field F_1, and since ϕ is not the zero map, we conclude that its kernel is trivial and that ϕ is a ring isomorphism. As ϕ fixes K, the image of E_1 under ϕ is $\phi[E_1] = \phi[K(x_1)] = K(\phi(x_1)) = K(\Gamma(x_2)) = K(x_2) = E_2$ by Theorem 2.3, so $\phi\big|_{E_1} : E_1 \to E_2$ is surjective and thus also a ring isomorphism. We conclude that ϕ is an isomorphism of superelliptic function fields. □

Proposition 3.12. *Let g_1 and g_2 be elements of $\mathcal{R}_m(K)$. Then (E_{g_1}, F_{g_1}) and (E_{g_2}, F_{g_2}) are isomorphic if and only if $g_1 = h^m \cdot \Gamma(g_2)^e$ for some $h \in K(X)^*$, $\Gamma \in \mathrm{PGL}_2(K)$, and $e \in \mathbb{Z}$ coprime to m.*

Proof. Suppose that we indeed have $g_1 = h^m \cdot \Gamma(g_2)^e$. Let $\psi : K(X)[Y] \to K(X)[Y]$ be the ring homomorphism given by fixing K and setting $\psi(X) = \Gamma^{-1}(X)$ and $\psi(Y) = \Gamma^{-1}(h(X))Y^e$, and let $\chi : K(X)[Y] \to F_{g_2}$ be the canonical quotient map. Then we have the following:

$$\begin{aligned}\psi(Y^m - g_1(X)) &= \psi(Y^m - h(X)^m \cdot \Gamma(g_2(X))^e) \\ &= \psi(Y)^m - h(\psi(X))^m \cdot \psi(\Gamma(g_2(X)))^e \\ &= \Gamma^{-1}(h(X))^m \cdot (Y^m)^e - \Gamma^{-1}(h(X))^m \cdot g_2(X)^e.\end{aligned}$$

Now we observe that $(\chi \circ \psi)(Y^m - g_1(X)) = 0$ since $\chi(Y^m - g_2(X)) = 0$. Therefore the ideal $(Y^m - g_1(X))$ is included in $\ker(\chi \circ \psi)$, and the homomorphism theorem for rings gives us a ring homomorphism $\phi : F_{g_1} \to F_{g_2}$ which fixes K and satisfies $\phi(X) = \Gamma^{-1}(X)$ and $\phi(Y_{g_1}) = \Gamma^{-1}(h(X))Y_{g_2}^e$. Finally Lemma 3.11 with the models (X, Y_{g_1}) and (X, Y_{g_2}) tells us that ϕ is an isomorphism of function fields.

Conversely, assume $(E_{g_1}, F_{g_1}) \cong (E_{g_2}, F_{g_2})$ with isomorphism $\phi : F_{g_1} \xrightarrow{\sim} F_{g_2}$. Lemma 3.11 with the models (X, Y_{g_1}) and (X, Y_{g_2}) shows us that $\phi(X) = \Gamma_0(X)$ and $\phi(Y_{g_1}) = h_0(X) \cdot Y_{g_2}^e$ for some $h_0 \in K(X)^*$, $\Gamma_0 \in \mathrm{PGL}_2(K)$, and $e \in \mathbb{Z}$ coprime to m. Now observe the following:

$$\begin{aligned}\Gamma_0(g_1(X)) &= \Gamma_0(Y_{g_1}^m) \\ &= (\Gamma_0 \circ \phi^{-1})(h_0(X)^m \cdot Y_{g_2}^{em}) \\ &= (\Gamma_0 \circ \phi^{-1})(h_0(X)^m \cdot g_2(X)^e) \\ &= h_0(X)^m \cdot g_2(X)^e\end{aligned}$$

We conclude that $g_1 = h^m \cdot \Gamma(g_2)^e$, where $h = \Gamma_0^{-1}(h_0)$ and $\Gamma = \Gamma_0^{-1}$. □

Proof of Theorem 3.7. Lemma 3.8 and Proposition 3.12 tell us that the given map is well-defined and independent of the choice of representative. The given inverse is independent of the choice of representative and model, since any two

resulting rational functions, say g_1 and g_2, will be elements of $\mathcal{R}_m(K)$ by Proposition 3.12, and will satisfy $(E_{g_1}, F_{g_1}) \cong (E_{g_2}, F_{g_2})$ by Lemma 3.10. Thus they are in the same orbit by Proposition 3.12. Finally, these maps are two-sided inverses, since for any $g \in \mathcal{R}_m(K)$, we may choose the model (X, Y_g) for (E_g, F_g) to retrieve g, and for any $(E, F) \in \mathcal{R}_m(K)$ with any model (x, y) and $g \in \mathcal{R}_m(K)$ such that $y^m = g(x)$, we have $(E, F) \cong (E_g, F_g)$ by Lemma 3.10. □

From an algorithmic perspective, Theorem 3.7 is still quite impractical for the purpose of enumerating isomorphism classes due to the size of $\mathcal{R}_m(K)$. However, any orbit of $\mathcal{R}_m(K)/(K(X)^*)^m$ contains an m^{th}-power free polynomial.

Definition 3.13. The set $\mathcal{P}_m(K)$ is the set consisting of the m^{th}-power free polynomials in $\mathcal{R}_m(K)$. In other words, these are the $g \in \mathcal{R}_m(K) \cap K[X]$ such that all irreducible factors of g have multiplicity less than m.

The choice of an m^{th}-power free polynomial as a representative for an orbit of $\mathcal{R}_m(K)/(K(X)^*)^m$ is unique up to a constant factor in $(K^*)^m$, and leads to a natural bijection between $\mathcal{R}_m(K)/(K(X)^*)^m$ and $\mathcal{P}_m(K)/(K^*)^m$. The inverse of this bijection is induced by the imbedding of $\mathcal{P}_m(K)$ in $\mathcal{R}_m(K)$. Using this bijection, we transfer the action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(K)$ to $\mathcal{P}_m(K)/(K^*)^m$. This proves the following.

Corollary 3.14 (of Theorem 3.7)**.** *Let K be a field containing a primitive m^{th} root of unity. Then there exists a bijection*

$$\left(\mathbb{Z}/m\mathbb{Z}\right)^* \times \mathrm{PGL}_2(K) \backslash \left(\mathcal{P}_m(K)/(K^*)^m\right) \xrightarrow{\sim} \mathcal{S}_m(K)/\cong$$

induced by the imbedding of $\mathcal{P}_m(K)$ in $\mathcal{R}_m(K)$ and the bijection of Theorem 3.7.

3.2 Binary Forms

In order to make the action of $\mathrm{PGL}_2(K)$ on $\mathcal{P}_m(K)/(K^*)^m$ more practical, we will instead consider the homogenisations of these polynomials, which admit a more intuitive action of $\mathrm{PGL}_2(K)$.

Definition 3.15. A *binary form* over K of degree $n \in \mathbb{Z}_{\geq 0}$ is a homogeneous bivariate polynomial over K of degree n, i.e., a polynomial in $K[X, Z]$ such that all monomials $X^i Z^j$ with non-zero coefficient satisfy $i + j = n$.

Definition 3.16. The *m-homogenisation* of a non-zero univariate polynomial $g \in K[X]$ is the binary form $G(X, Z) = Z^n g\left(\frac{X}{Z}\right)$, where $n = \min\{t \in m\mathbb{Z} : t \geq \deg g\}$. The *dehomogenisation* of a binary form $G \in K[X, Z]$ is the univariate polynomial $g(X) = G(X, 1)$.

Evidently factorisation is preserved under (de)homogenisation, apart from powers of Z. This means that that binary forms factor into irreducible binary forms, and that the irreducible binary forms are Z and those binary forms not divisible by Z of which the dehomogenisation is irreducible.

Definition 3.17. A binary form is *monic* if its dehomogenisation is monic.

Remark 3.18. This definition is not standard in literature, and only serves as our default choice of a binary form up to scalar multiplication.

Definition 3.19. Let $\mathcal{B}_m(K)$ be the set of non-constant binary forms $G \in K[X,Z]$ of the form $G(X,Z) = k \cdot \prod_{i=1}^{s} G_i(X,Z)^{n_i}$ for some $k \in K^*$, positive $s, n_i \in \mathbb{Z}$, and distinct monic irreducible binary forms $G_i \in K[X,Z]$ satisfying $m \mid \deg G$, $n_i < m$ for all $1 \leq i \leq s$, and $\gcd(m, n_1, \ldots, n_s) = 1$.

Proposition 3.20. *The process of m-homogenisation defines a bijection from $\mathcal{P}_m(K)$ to $\mathcal{B}_m(K)$, and its inverse is given by dehomogenisation. The induced left action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(K)$ on $\mathcal{B}_m(K)/(K^*)^m$ is as follows: let $(\overline{e}, \Gamma) \in (\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(K)$ and $G \in \mathcal{B}_m(K)$, and write $G(X,Z) = k \cdot \prod_{i=1}^{s} G_i(X,Z)^{n_i}$ as in Definition 3.19, then $(\overline{e}, \Gamma)$ sends the orbit of G to the orbit of*

$$k^e \cdot \prod_{i=1}^{s} G_i(dX - bZ, -cX + aZ)^{(en_i \text{ rem } m)},$$

where $e \in \mathbb{Z}$ and $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{GL}_2(K)$ are representatives of $\overline{e}$ and Γ, and $(n \text{ rem } m)$ denotes the non-negative remainder of the integer division of n by m.

Proof. The m-homogenisation G of a rational function $g \in \mathcal{P}_m(K)$ is in $\mathcal{B}_m(K)$, because m-homogenisation preserves the factorisation of g apart from introducing the factor Z with multiplicity $\deg G - \deg g < m$. Conversely, the dehomogenisation g of a binary form $G \in \mathcal{B}_m(K)$ is in $\mathcal{P}_m(K)$. After all, let d be the greatest common divisor of m and the multiplicities of the irreducible factors of g, then $d \mid (\deg G - \deg g)$ because $d \mid \deg G$. Additionally, dehomogenisation and m-homogenisation are clearly two-sided inverses, because m-homogenisation ensures we choose the correct degree when homogenising.

The given action is independent of the choice of representative for $\overline{e}$ and Γ, since for any $\lambda \in K^*$ and binary form H we have $H(\lambda S, \lambda T) = \lambda^{\deg H} H(S,T)$, and $m \mid \sum_{i=1}^{s} (en_i \text{ rem } m)$ because $m \mid \deg G$. One immediately checks that

$$G_i(dX - b, -cX + a) = (-cX + a)^{\deg G_i} \cdot G_i\left(\frac{dX - b}{-cX + a}, 1\right)$$

is an m^{th}-power free representative of $G_i\left(\frac{dX-b}{-cX+a}, 1\right)$, and that it is also the dehomogenisation of $G_i(dX - bZ, -cX + aZ)$. The given image is clearly an m^{th}-power free polynomial in the $(K(X)^*)^m$-orbit of $G(dX - bZ, -cX + aZ)^e$, so the given action matches the induced action. □

Theorem 3.21. *Let K be a field containing a primitive m^{th} root of unity. Then there exists a bijection*

$$\left(\mathbb{Z}/m\mathbb{Z}\right)^* \times \mathrm{PGL}_2(K) \backslash \left(\mathcal{B}_m(K)/(K^*)^m\right) \xrightarrow{\sim} \mathcal{S}_m(K)/\cong$$

induced by dehomogenisation and the bijection of Corollary 3.14.

Proof. This follows directly from Corollary 3.14 and Proposition 3.20. □

In our algorithm we will mostly concern ourselves with the irreducible factors of binary forms, and we would rather not keep track of constant factors. We may forget these constant factors using the natural surjection from $\mathcal{B}_m(K)/(K^*)^m$ to $\mathcal{B}_m(K)/K^*$. Given two elements of $\mathcal{B}_m(K)/(K^*)^m$ that are in the same class modulo K^*, their images under the action described in Proposition 3.20 clearly also belong to the same class modulo K^*. Therefore we may also induce an action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(K)$ on $\mathcal{B}_m(K)/K^*$.

Corollary 3.22. *Let K be a field containing a primitive m^{th} root of unity. Then there exists a surjection*

$$\mathcal{S}_m(K)\Big/\cong \quad\longrightarrow\quad \left(\mathbb{Z}/m\mathbb{Z}\right)^* \times \mathrm{PGL}_2(K) \Big\backslash \left(\mathcal{B}_m(K)\Big/K^*\right)$$

induced by the isomorphism of Theorem 3.21 and the natural surjection $\chi : \mathcal{B}_m(K)/(K^*)^m \to \mathcal{B}_m(K)/K^*$. Any element of the codomain has at most m pre-images, and these are given by the isomorphism classes of the superelliptic function fields $\left(E_{k\cdot G(X,1)}, F_{k\cdot G(X,1)}\right)$, where $G \in \mathcal{B}_m(K)$ is a representative of said element of the codomain and $k \in K^*$ ranges over representatives of $K^*/(K^*)^m$.

Remark 3.23. The given pre-images might not all be distinct, since some of the given superelliptic function fields might be isomorphic. These function fields are typically referred to as each other's *m^{th}-power twists*. More generally, twists are algebraic function fields that would be isomorphic if their constant field was replaced by its algebraic closure.

Proof. This follows directly from Theorem 3.21 because we defined the action of $\left(\mathbb{Z}/m\mathbb{Z}\right)^* \times \mathrm{PGL}_2(K)$ on $\mathcal{B}_m(K)/K^*$ through χ, and the given pre-images correspond to the fibres of χ. □

3.3 Projective Linear Group

In this section we will study the action of $\mathrm{PGL}_2(\mathbb{F}_q)$ on binary forms through their zeroes, which will prove to be a convenient approach for Section 4. We will consider the action of $\Gamma \in \mathrm{PGL}_2(K)$ on arbitrary binary forms $G \in K[X,Z]$ up to scaling by $(K^*)^m$, by which we mean

$$\Gamma(G \mod (K^*)^m) = G(dX - bZ, -aX + cZ) \mod (K^*)^m,$$

in line with Proposition 3.20. This approach also allows us to estimate the number of isomorphism classes of superelliptic function fields in Theorem 3.33.

Definition 3.24. The *projective line* $\mathbb{P}^1(K)$ over a field K is the set $K^2\backslash\{(0,0)\}$ up to scalar multiplication by K^*. The class of $(a,b) \in K^2 \setminus \{(0,0)\}$ in $\mathbb{P}^1(K)$ is denoted $(a:b)$.

The projective linear group $\mathrm{PGL}_2(K)$ acts on the projective line through matrix-vector multiplication.

Definition 3.25. The zero set of a binary form $G \in K[X,Z]$ is the set

$$\mathrm{zeroes}(G) := \{(a:b) \in \mathbb{P}^1(\overline{K}) : G(a,b) = 0\}$$

Note that this set is well-defined because G is homogeneous. If we consider the action of $\mathrm{PGL}_2(K)$ on $\mathcal{B}_m(K)/(K^*)^m$, as in Proposition 3.20 with $e = 1$, then we observe that $\Gamma \in \mathrm{PGL}_2(K)$ maps the zeroes of G to the zeroes of $\Gamma(G)$.

Lemma 3.26 (cf. [Aud02, Prop. V.5.6])**.** *Let (a_0, a_1, a_2) and (b_0, b_1, b_2) be two triples of distinct elements of $\mathbb{P}^1(K)$. Then there exists a unique $\Gamma \in \mathrm{PGL}_2(K)$ such that $\Gamma(a_i) = b_i$ for $i = 0, 1, 2$.*

Corollary 3.27. *The group $\mathrm{PGL}_2(\mathbb{F}_q)$ has order $q^3 - q$.*

Proof. Note that $\mathbb{P}^1(\mathbb{F}_q)$ has $q+1 \geq 3$ elements. Lemma 3.26 tells us that there are precisely as many elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ as there are ordered triples of distinct elements of $\mathbb{P}^1(\mathbb{F}_q)$. □

Proposition 3.28. *Let $G, H \in K[X,Z]$ be two binary forms with $n \geq 3$ distinct zeroes in $\mathbb{P}^1(\overline{K})$. Then the set $\{\Gamma \in \mathrm{PGL}_2(K) : \Gamma(G) \equiv H \pmod{K^*}\}$ contains at most $n \cdot (n-1) \cdot (n-2)$ elements.*

Proof. This follows directly from Lemma 3.26 and the observation that any element of $\{\Gamma \in \mathrm{PGL}_2(K) : \Gamma(G) \equiv H \pmod{K^*}\}$ is an element of $\mathrm{PGL}_2(\overline{K})$ that maps three distinct zeroes of G to three distinct zeroes of H. □

Definition 3.29. Let $G \in K[X,Z]$ be an m^{th}-power free binary form. The *square-free factorisation* of G is the unique tuple of pairwise coprime monic binary forms $(G_i)_{1 \leq i < m}$ such that $G = k \cdot \prod_{i=1}^{m-1} G_i^i$ for some $k \in K^*$.

Definition 3.30. A *square-free factorisation type (of exponent m)* D is a tuple of non-negative integers $(d_i)_{1 \leq i < m}$. The square-free factorisation type of an m^{th}-power free binary form $G \in K[X,Z]$ is the tuple $(\deg G_i)_{1 \leq i < m}$, where $(G_i)_{1 \leq i < m}$ is the square-free factorisation of G. The *order* of D is $\sum_{i=1}^{m-1} d_i$.

Remark 3.31. The order of a square-free factorisation type is the number of distinct zeroes in $\mathbb{P}^1(K)$ of any binary form of that type, and not its degree, which would be $\sum_{i=1}^{m-1} i \cdot d_i$.

We say that D is a *valid* square-free factorisation type for $\mathcal{B}_m(\mathbb{F}_q)$ if m divides $\sum_{i=1}^{m-1} i \cdot d_i$ and the greatest common divisor of m and all elements of $\{i \in \{1,2,\ldots,m\} : d_i \neq 0\}$ is one, which means binary forms of this type satisfy the conditions of Definition 3.19.

The square-free factorisation type is invariant under the action of $\mathrm{PGL}_2(K)$, and the action of $\overline{e} \in (\mathbb{Z}/m\mathbb{Z})^*$ permutes $(m-1)$-tuples, such as the square-free factorisation (type), as follows: the image of $(s_i)_{1 \leq i < m}$ under $\overline{e}$ is the tuple $(t_i)_{1 \leq i < m}$ with $t_{(e \cdot i \bmod m)} = s_i$. We say that two square-free factorisation types are *equivalent* if they can be permuted into each other in this way.

Definition 3.32. Let D be a square-free factorisation type of exponent m that is valid for $\mathcal{B}_m(\mathbb{F}_q)$. The set $\mathcal{B}_m^D(\mathbb{F}_q)$ is the set of $G \in \mathcal{B}_m(\mathbb{F}_q)$ with square-free factorisation type equivalent to D.

Note that we can restrict the action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ on $\mathcal{B}_m(\mathbb{F}_q)/\mathbb{F}_q^*$ to $\mathcal{B}_m^D(\mathbb{F}_q)\Big/\mathbb{F}_q^*$, because $(\mathbb{Z}/m\mathbb{Z})^*$ keeps the square-free factorisation type equivalent to D, and $\mathrm{PGL}_2(\mathbb{F}_q)$ does not affect the square-free factorisation type.

Theorem 3.33. *Let D be a square-free factorisation type of order $n \geq 3$ that is valid for $\mathcal{B}_m(\mathbb{F}_q)$. If we take m and n to be fixed, then the number of orbits of $\mathcal{B}_m^D(\mathbb{F}_q)\Big/\mathbb{F}_q^*$ under the action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ is $\Theta(q^{n-3})$.*

Proof. The group $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ has $\Theta(q^3)$ elements by Corollary 3.27, and there are $\Theta(q^n)$ elements of $\mathcal{B}_m^D(\mathbb{F}_q)\Big/\mathbb{F}_q^*$, because there are already $\binom{q}{n}$ such monic binary forms that split into linear factors, and $\binom{q}{n}$ is $\Theta(q^n)$ for fixed n.

Proposition 3.28 tells us that the stabiliser of any element of $\mathcal{B}_m^D(\mathbb{F}_q)\Big/\mathbb{F}_q^*$ has $O(mn^3)$ elements, which is constant in q. Thus the orbit of any element of $\mathcal{B}_m^D(\mathbb{F}_q)\Big/\mathbb{F}_q^*$ is of size $\Theta(q^3)$, and there are $\Theta(q^{n-3})$ orbits in total. □

4 Algorithms

In Section 4, we will discuss the algorithms we implemented in Julia. We will start with several algorithms by Everett Howe [How25a; How25b] in Section 4.1. These serve both as building blocks and inspiration for our own algorithm, which will be discussed in Section 4.2.

Throughout this section, we measure our complexity as a function of the field size q alone, taking all other parameters to be fixed, and we assume that we have separate total orderings on $\mathbb{F}_q$, $\mathbb{F}_q[X]$, and the homogeneous polynomials of $\mathbb{F}_q[X, Z]$. We will discuss the implementation of such orderings in Section 5.1.

We will often need to find elements of the projective linear group that send one binary form to another. Explicit methods to construct such elements can be found in Appendix A.

4.1 Enumerating Hyperelliptic Function Fields

Our work is motivated by Everett Howe's algorithm for enumerating isomorphism classes of *hyperelliptic function fields*, which are superelliptic function fields of exponent $m = 2$ [How25a]. In this case, the group $(\mathbb{Z}/m\mathbb{Z})^*$ is trivial, so the isomorphisms of hyperelliptic function fields, as defined in Definition 3.4, are given by the action of $\mathrm{PGL}_2(\mathbb{F}_q)$ alone. The original paper only deals with fields of odd characteristic, but in a follow-up paper, Howe also presents an algorithm for enumerating $\mathrm{PGL}_2(\mathbb{F}_q)$ orbits of square-free binary forms that works in any non-zero characteristic and has improved memory complexity [How25b].

4.1.1 Structure of Howe's Algorithms

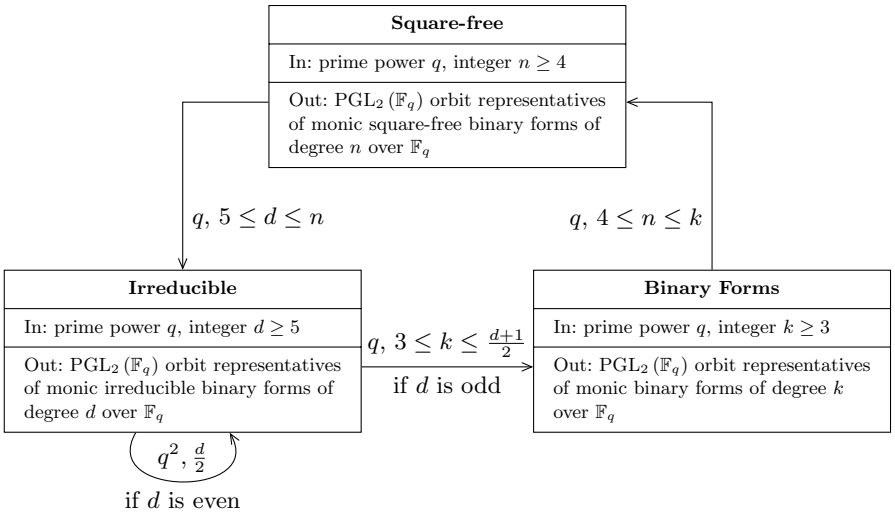

Figure 1: The recursive structure of Howe's algorithm for enumerating hyperelliptic function fields. The notation $A \xrightarrow{q,\ r \leq s \leq t} B$ means that algorithm A calls algorithm B for the prime power q and all integers $s \in \{r, r+1, \ldots, t\}$.

Figure 1 provides an overview of the recursive structure of Howe's algorithm. Instead of the set $\mathcal{B}_2(\mathbb{F}_q)/\mathbb{F}_q^*$, he enumerates monic square-free binary forms, which may be seen as the monic representatives of $\mathcal{B}_2(\mathbb{F}_q)/\mathbb{F}_q^*$. Howe additionally provides non-recursive methods to enumerate of smaller square-free and irreducible binary forms [How25a, Theorems 1.2 and 3.8].

The time complexity of these algorithms, measuring the number of arithmetic operations $\mathbb{F}_q$ as a function of q for fixed n, d, and k, is $\widetilde{O}(q^{n-3})$, $\widetilde{O}(q^{d-3})$, and $\widetilde{O}(q^{k-3})$ for *Square-free*, *Irreducible*, and *Binary Forms* respectively. All three algorithms require $O(\log q)$ memory.

Our own algorithm is based on the *Square-free* algorithm, and this algorithm will be discussed in detail in Section 4.1.2. We will now provide a short overview of the other two algorithms, since we do need them for our implementation.

The *Irreducible* algorithm of Figure 1 consists of two algorithms for enumerating irreducible binary forms of degree $d \geq 5$ over $\mathbb{F}_q$. The first enumerates handles even d, and involves enumerating irreducible binary forms over $\mathbb{F}_{q^2}$ of degree $\frac{d}{2}$ up to $\mathrm{PGL}_2\left(\mathbb{F}_{q^2}\right)$, and finding coset representatives for $\mathrm{PGL}_2\left(\mathbb{F}_q\right) \subset \mathrm{PGL}_2\left(\mathbb{F}_{q^2}\right)$. More details can be found in [How25b, Section 7].

A second algorithm handles odd d. Howe associates with every irreducible binary form of degree d a certain binary form of degree at most $\frac{d+1}{2}$ called the *Frobenius divisor*, in a way that behaves well with respect to the action of $\mathrm{PGL}_2\left(\mathbb{F}_q\right)$ [How25b, Section 2]. The algorithm revolves around finding all irreducible binary forms associated with a given Frobenius divisor, which, crucially for performance, involves factoring binary forms of degree $\left(\frac{d-1}{2}\right)^d$. More details can be found in [How25b, Section 6].

The *Binary forms* algorithm of Figure 1 is perhaps the simplest of the three. Given a binary form, we may consider its square-free part, i.e., the product of its distinct irreducible factors. If two binary forms are in the same orbit, then so are their square-free parts. The algorithm uses orbit representatives for square-free binary forms, factors them, and tries out different multiplicities for those factors. More details may be found in [How25b, Section 5.2].

4.1.2 Enumerating Square-free Binary Forms

Definition 4.1. Let G be a square-free binary form over the field $\mathbb{F}_q$. Then the *Galois type* of G is the tuple of the degrees of the irreducible factors of G, sorted non-ascendingly. The *degree* of a Galois type is the sum of its constituents, i.e., the degree of any binary form of that type.

We will also consider Galois types without an explicit square-free binary form G, in which case we just mean a non-ascending tuple of positive integers. Howe starts by observing that the action of $\mathrm{PGL}_2\left(\mathbb{F}_q\right)$ does not affect the Galois type, and that we can therefore consider one Galois type at a time.

Lemma 4.2. *Let M be a Galois type of degree 3 or lower. Then the action of $\mathrm{PGL}_2\left(\mathbb{F}_q\right)$ on the binary forms up to scaling by $\mathbb{F}_q^*$ of type M is transitive.*

Proof. If M only consists of linear factors, then Lemma 3.26 proves the statement. Simple counting arguments show that there are $\frac{q^2-q}{2} \cdot (q+1) = \frac{q^3-q}{2}$ monic binary forms of Galois type $(2,1)$, and the stabiliser of such a binary form has at most 2 elements, because we can either swap the zeroes of the quadratic binary form or leave them in place. Similarly, there are $\frac{q^3-q}{3}$ monic binary forms of

Galois type (3), and the stabiliser of such a binary form has at most 3 elements by Corollary A.2. For both types, we can conclude that there can only be one orbit, because Corollary 3.27 tells us that $\mathrm{PGL}_2(\mathbb{F}_q)$ has $q^3 - q$ elements. □

Representatives for the type $(2, 1, 1)$ are given in [How25a, Thm. 3.6], and algorithms for enumerating the other Galois types $(m_1, m_2, \ldots, m_r)$ of degree at least 4 are given in [How25b, Section 5.1] according to the following case distinction on the Galois type of the output:

1. $m_1 \geq 3$ [How25b, Algorithm 5.1];
2. $m_1 = m_2 = 2$ [How25b, Algorithm 5.4];
3. $m_1 \leq 2$, $m_2 = 1$, and at least three of the m_i are 1 [How25b, Algorithm 5.6].

Let us consider the first of these three algorithms. We will only prove its correctness, and refer the reader to Howe's paper for the complexity analysis.

Definition 4.3 (cf. [How25a, Def. 4.1])**.** Let G be a monic irreducible binary form of degree $d \geq 4$ over the field $\mathbb{F}_q$ and let $\alpha \in \mathbb{F}_{q^d}$ be a root of $G(X, 1)$. The *cross polynomial* of G, notated $\mathrm{cross}(G)$, is the characteristic polynomial over $\mathbb{F}_q$ of

$$\chi := \frac{(\alpha^{q^3} - \alpha^q)(\alpha^{q^2} - \alpha)}{(\alpha^{q^3} - \alpha)(\alpha^{q^2} - \alpha^q)} \in \mathbb{F}_{q^d}.$$

Lemma 4.4 (cf. [How25a, Thm. 4.2])**.** *Two monic irreducible binary forms G and H of degree $d \geq 4$ over the field $\mathbb{F}_q$ are in the same $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit if and only if $\mathrm{cross}(G) = \mathrm{cross}(H)$.*

Algorithm 4.5 (cf. [How25b, Algorithm 5.1])**.** $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit representatives of monic square-free binary forms of Case 1.

Input: A finite field $\mathbb{F}_q$, an integer $n \geq 4$, and an integer d such that $3 \leq d \leq n$.

Output: $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit representatives of the monic binary forms over $\mathbb{F}_q$ of every Galois types $(m_1, m_2, \ldots, m_r)$ such that $m_1 = d$.

1. For every $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit representative P of degree d irreducible binary forms over $\mathbb{F}_q$, do:
   (a) For every monic square-free binary form G over $\mathbb{F}_q$ of degree $n - d$ satisfying the following:
      (x) P does not divide G;
      (y) G has no irreducible factor of degree larger than d;
      (z) If $d \geq 4$, then G has no irreducible factor Q of degree d with $\mathrm{cross}(Q) < \mathrm{cross}(P)$;

      do:
      (i) Let $H := P \cdot G$.
      (ii) Determine the set $S := \{\Gamma \in \mathrm{PGL}_2(\mathbb{F}_q) : P \mid \Gamma(H)\}$. Appendix A shows us how to find the Γ that map an irreducible degree d factor of H with the same cross polynomial as P to P.

(iii) Output H if $H = \min_{\Gamma \in S}\{\Gamma(H)\}$, and continue with the next value of G.

Remark 4.6. In his paper, Howe instead gives an algorithm that enumerates binary forms of one specific Galois type with $m_1 = d$. We group all Galois types with $m_1 = d$ together, since this reduces the number of times we enumerate the degree d irreducible binary forms.

Proposition 4.7. *Algorithm 4.5 produces the correct output.*

Proof. Let us consider the values that H will take in Step 1(a)(i). These are precisely the monic binary forms $H \in \mathcal{B}_2(\mathbb{F}_q)$ of Case 1 satisfying the following:

1. H has no irreducible factors of degree larger than d by Condition 1(a)(y);
2. H has an irreducible factor P of degree d such that P is one of the representatives considered in Step 1;
3. Any other irreducible factor Q of H of degree d either belongs to the same $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit as P, or has degree at least 4 and cross polynomial larger than P, due to Condition 1(a)(z). Note that P and Q always belong to the same orbit if $d = 3$ due to Lemma 4.2.

The set S of Step 1(a)(ii) now consists of all elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ that map such an H to another element of the orbit of H that satisfies the above properties. Therefore, Step 1(a)(iii) ensures we do not output multiple binary forms from the same orbit. Additionally, given a monic binary form with an irreducible factor of degree d and no larger degree irreducible factors, we can use the action of $\mathrm{PGL}_2(\mathbb{F}_q)$ to move one of the degree d irreducible factors with minimal cross polynomials to one of the representatives considered in Step 1. □

The algorithm for Case 2 works nearly identically, but uses orbit representatives for binary forms of Galois type $(2, 2)$ in Step 1, given by [How25a, Theorems 3.4 and 3.8]. Instead of the cross polynomial, it uses the following $\mathrm{PGL}_2(\mathbb{F}_q)$ invariant for binary forms of Galois type $(2, 2)$, which we will also use in Algorithms 4.16 and 4.19.

Definition 4.8 (cf. [How25b, Section 2.6])**.** Let $G \in \mathbb{F}_q[X, Z]$ be a monic square-free binary form of Galois type $(2, 2)$ that factors into the irreducible binary forms $G_1(X, Z) = X^2 + sXZ + tZ^2$ and $G_2(X, Z) = X^2 + uXZ + vtZ^2$. Then

$$\lambda(G) := \frac{(s-u)(sv-tu) + (t-v)^2}{(s^2-4t)(u^2-4v)}.$$

Lemma 4.9 (cf. [How25b, Lemma 2.16])**.** *Two monic square-free binary forms G and H over $\mathbb{F}_q$ of Galois type $(2, 2)$ are in the same $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit if and only if $\lambda(G) = \lambda(H)$.*

Finally, the algorithm for Case 3 fixes three linear factors to Z, X, and $X - Z$, and considers all elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ that map another triple of linear factors to $XZ(X - Z)$. This is sufficient since the action of $\mathrm{PGL}_2(\mathbb{F}_q)$ is transitive on Galois type $(1, 1, 1)$ by Lemma 4.2.

4.2 Enumerating Superelliptic Function Fields

In this section, we will provide an algorithm that proves the following theorem.

Theorem 4.10. *There exists an algorithm that takes as input a finite field $\mathbb{F}_q$, an integer $m > 2$ that divides $q-1$, and a square-free factorisation type of exponent m and order $n \geq 3$ that is valid for $\mathcal{B}_m(\mathbb{F}_q)$ as in Definition 3.30, and gives exactly one representative of every orbit of $\mathcal{B}_m^D(\mathbb{F}_q)\big/(\mathbb{F}_q^*)^m$ under the action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ as described in Proposition 3.20, which runs in time $\widetilde{O}(q^{n-3})$ if $n > 3$ and $\widetilde{O}(\sqrt{q})$ if $n = 3$, and takes $O(\log q)$ memory.*

4.2.1 From Binary Forms to Function Fields

Recall that Corollary 3.22 shows us the following. Given representatives for the orbits of $\mathcal{B}_m(\mathbb{F}_q)/\mathbb{F}_q^*$ under the action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$, we can take all of these representatives $(G \mod \mathbb{F}_q^*)$, and take all their multiples $(k \cdot G \mod (\mathbb{F}_q^*)^m)$ for k in a set of representatives for $\mathbb{F}_q^*/(\mathbb{F}_q^*)^m$, to get a complete, but possibly duplicate, set of representatives for the orbits of $\mathcal{B}_m(\mathbb{F}_q)/(\mathbb{F}_q^*)^m$ under the action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$. The action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ is essentially the same on both of these sets apart from how we handle constant factors, and is defined in Proposition 3.20.

In this section, we will discuss how to find a set of representatives $C \subset \mathbb{F}_q^*$ for $\mathbb{F}_q^*/(\mathbb{F}_q^*)^m$, and how to find a maximal subset $A \subset C$ such that all $(a \cdot G \mod (\mathbb{F}_q^*)^m)$ for $a \in A$ belong to different $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ orbits. Then Theorem 3.21 shows us how we may use these representatives of $\mathcal{B}_m(\mathbb{F}_q)/(\mathbb{F}_q^*)^m$ to enumerate superelliptic function fields.

First of all, we need to determine a set of representatives in $\mathbb{F}_q^*$ for $\mathbb{F}_q^*/(\mathbb{F}_q^*)^m$. We can do this by finding a *primitive element* of $\mathbb{F}_q$, by which we mean a generator γ of the cyclic multiplicative group $\mathbb{F}_q^*$. Given such a γ, the set $C := \{\gamma^i : 0 \leq i < m\}$ is a set of representatives for $\mathbb{F}_q^*/(\mathbb{F}_q^*)^m$. In practice, we find such a γ by using the probabilistic OSCAR function `Hecke.primitive_element` from the Julia package Hecke [Fie+17], but this can also be done deterministically in time $O(q^{1/4+\epsilon})$ for any $\epsilon > 0$ using the methods described in [Shp96].

Suppose that we are given a $(G \mod \mathbb{F}_q^*) \in B_m(\mathbb{F}_q)/\mathbb{F}_q^*$, along with its $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ stabiliser S. The elements of S only stabilise G up to $\mathbb{F}_q^*$, and if we apply the elements of S to $G \mod (\mathbb{F}_q^*)^m$ instead, we will get $k \cdot G \mod (\mathbb{F}_q^*)^m$ for some $k \in \mathbb{F}_q^*$. The following lemma tells us how to determine the $\mathbb{F}_q^*/(\mathbb{F}_q^*)^m$ orbit of such k.

Lemma 4.11. *Let $\alpha, \beta \in \mathbb{F}_q^*$. Then α and β are in the same class of $\mathbb{F}_q^*/(\mathbb{F}_q^*)^m$ if and only if $\alpha^{\frac{q-1}{m}} = \beta^{\frac{q-1}{m}}$.*

Proof. Let δ be $\frac{\alpha}{\beta}$. Then $\delta^{\frac{q-1}{m}} = \frac{\alpha^{\frac{q-1}{m}}}{\beta^{\frac{q-1}{m}}}$ is 1 if and only if δ is an m^{th} power. □

Algorithm 4.12. Enumerating $\mathbb{F}_q^*/(\mathbb{F}_q^*)^m$ using representatives of $\mathbb{F}_q^*/\mathbb{F}_q^*$.

Input: A finite field $\mathbb{F}_q$, an integer $m \geq 2$ that divides $q-1$, a set $C \subset \mathbb{F}_q^*$ of representatives for $\mathbb{F}_q^*/(\mathbb{F}_q^*)^m$, a $(G \mod \mathbb{F}_q^*) \in B_m(\mathbb{F}_q)/\mathbb{F}_q^*$ with square-free factorisation type of order at least 3, and the $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ stabiliser S of $(G \mod \mathbb{F}_q^*)$.

Output: A maximal subset $A \subset C$ such that all $(k \cdot G \mod (\mathbb{F}_q^*)^m)$ for $k \in A$ belong to different $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ orbits.

1 Let A be the empty set.

2 While C is not empty, do:

   (a) Choose an element k of C, and add k to A.

   (b) For every $(\overline{e}, \Gamma) \in S$, do:

      i. Let $u \in \mathbb{F}_q^*$ be such that $\Gamma(k \cdot G \mod (\mathbb{F}_q^*)^m)^{\overline{e}} = u \cdot G \mod (\mathbb{F}_q^*)^m$.

      ii. If C contains an element c with the same class modulo $(\mathbb{F}_q^*)^m$ as $\frac{u}{k}$, which we determine using Lemma 4.11, then remove c from C.

3 Return A.

Proposition 4.13. *Algorithm 4.12 is correct and runs in time $\widetilde{O}(\log q)$ and $O(1)$ memory.*

Proof. Note that Algorithm 4.12 terminates, because the identity is an element of S, so in every iteration of Step 2, we will remove the k chosen in Step 2(a) in Step 2(b)(ii). The algorithm is correct, because in Step 2(b)(ii) we remove precisely the $c \in C$ such that $(c \cdot G \mod (\mathbb{F}_q^*)^m)$ is in the same orbit as $(k \cdot G \mod (\mathbb{F}_q^*)^m)$, and therefore we only ever choose k in Step 2(a) such that $(k \cdot G \mod (\mathbb{F}_q^*)^m)$ is not in the same orbit as any of the $(a \cdot G \mod (\mathbb{F}_q^*)^m)$ for $a \in A$.

The sets A and C both have $O(m)$ elements, which is constant in terms of q. The set S also has a constant number of elements in terms of q due to Proposition 3.28 and the fact that the square-free factorisation type of G has order at least 3. We can apply Lemma 4.11 in $O(\log q)$ multiplications in $\mathbb{F}_q^*$ using repeated squaring [GG99, Algorithm 4.3]. Therefore the algorithm runs in time $O(\log q)$ and $O(1)$ memory. □

4.2.2 Enumerating Monic Binary Forms

Now it suffices to enumerate the orbits of $\mathcal{B}_m^D(\mathbb{F}_q) \big/ \mathbb{F}_q^*$ under the left action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$. We will also determine the stabilisers of these orbit representatives, as we need them for Section 4.2.1. In the remainder of this section, we will identify $\mathcal{B}_m^D(\mathbb{F}_q) \big/ \mathbb{F}_q^*$ with the monic representative of each class.

Our algorithm consists of 5 algorithms, according to the following partition of the output:

A There is an irreducible factor of degree 3 or larger (Algorithm 4.14).

B There are two irreducible factors of degree 2 of the same multiplicity, and there are no larger irreducible factors (Algorithm 4.16).

C There are two irreducible quadratic factors of the different multiplicities, there is at most one quadratic factor of any multiplicity, and there are no larger irreducible factors (Algorithm 4.19).

D There is precisely one multiplicity with an irreducible quadratic factor, and all other irreducible factors are linear (Algorithm 4.21).

E All irreducible factors are linear (Algorithm 4.23).

In our algorithms, we will often construct our representatives by first choosing a couple of relatively small factors from $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit representatives provided by Howe, and then considering all binary forms we could multiply these factors with to obtain a binary form of square-free factorisation type D. Given the degrees and multiplicities of those chosen factors, we can determine the square-free factorisation type D' that this remainder must have. This is what we mean when we say 'the square-free factorisation type left after choosing P_1 with multiplicity m_1, P_2 with multiplicity m_2, ...'.

Case 4.14 mimics Algorithm 4.5 by Howe, enumerating binary forms with an irreducible factor of sufficiently large degree, and using the cross polynomial, as in Definition 4.3, to identify the $\mathrm{PGL}_2(\mathbb{F}_q)$ orbits of any other factors of that same degree. However, we also need to consider the multiplicity of these large factors.

Algorithm 4.14. Enumerating monic m^{th}-power free binary forms of Case A.

Input: A finite field $\mathbb{F}_q$, an integer $m \geq 2$ that divides $q-1$, an integer $k \geq 3$, and a valid square-free factorisation type $D = (d_i)_{1 \leq i \leq m-1}$ such that at least one d_i is equal to or larger than k.

Output: Orbit representatives and their stabilisers for the monic binary forms of $\mathcal{B}_m^D(\mathbb{F}_q)$ with an irreducible factor of degree k and no irreducible factors of larger degree.

1. Determine the group E consisting of the $\overline{e} \in (\mathbb{Z}/m\mathbb{Z})^*$ that fix D under permutation, as defined in Section 3.3.
2. Determine a complete set of representatives M of $\{1, 2, \ldots, m-1\}$ under multiplication by E modulo m.
3. For every $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit representative P of degree k irreducible binary forms over $\mathbb{F}_q$ given by [How25b, Algorithms 6.1 and 7.1], and every $m_P \in M$ such that $d_{m_P} \geq k$, do:
   (a) Let D' be the square-free factorisation type left after choosing P with multiplicity m_P.
   (b) For every monic m^{th}-power free binary form G over $\mathbb{F}_q$ of square-free factorisation type D' satisfying the following:
      (x) P does not divide G;
      (y) G has no irreducible factor of degree larger than k;
      (z) If $k \geq 4$, then G has no irreducible factor Q of degree k such that $\mathrm{cross}(Q) < \mathrm{cross}(P)$;

      do:

      (i) Let $H := P^{m_P} \cdot G$.
      (ii) Determine the set S of $(\overline{e}, \Gamma) \in E \times \mathrm{PGL}_2(\mathbb{F}_q)$ such that P divides $\Gamma(H)^{\overline{e}}$ and its multiplicity is an element of M. Appendix A shows us how to find the Γ that map an irreducible degree k factor of H with the same cross polynomial as P to P.

(iii) Output H and its stabiliser if $H = \min_{(\overline{e},\Gamma)\in S}\{\Gamma(H)^{\overline{e}}\}$, and continue with the next value of G. The stabiliser of H is contained in S and is found while applying the elements of S to H.

Proposition 4.15. *Algorithm 4.14 is correct and runs in time* $\widetilde{O}(q^{n-3})$ *and* $O(\log q)$ *memory, where* $n = \sum_{i=1}^{m-1} d_i$.

Proof. Let us consider the values that H in Step 3(b)(ii) will take. These are precisely the monic binary forms $H \in \mathcal{B}_m^D(\mathbb{F}_q)$ of Case A satisfying the following:

1. H has square-free factorisation type D, by Step 3(b);
2. H has an irreducible factor P of degree k such that P is one of the representatives considered in Step 3, and the multiplicity of P is an element of M, due to Step 3;
3. Any other irreducible factor Q of H of degree k either belongs to the same $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit as P, or has degree at least 4 and cross polynomial larger than P, due to Condition 3(b)(z). Note that P and Q always belong to the same orbit if $k = 3$ due to Lemma 4.2.

The set S of Step 3(b)(ii) now consists of all elements of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ that map such an H to another element of the orbit of H that satisfies the above properties. Therefore, Step 3(b)(iii) ensures we do not output multiple binary forms from the same orbit.

We still need to show that every orbit of $\mathcal{B}_m^D(\mathbb{F}_q)\big/\mathbb{F}_q^*$ of Case A contains a binary form H satisfying the above properties. Given an element G of such an orbit, we can use $(\mathbb{Z}/m\mathbb{Z})^*$ to move it to an element with square-free factorisation type D. Additionally, G has an irreducible factor P of degree k that has minimal cross polynomials amongst all irreducible factors of degree k, and we can move it to a multiplicity in M using E. Finally, the action of $\mathrm{PGL}_2(\mathbb{F}_q)$ can move P to one of the representatives considered in Step 3.

Let us consider the set S. If P divides $\Gamma(H)^{\overline{e}}$, then there is an irreducible factor Q of H of degree k such that $\Gamma(Q) = P$, and is thus in the same orbit as P. These Q can already be determined in Step 3(b)(z), and there are $O(\frac{n}{k})$ such Q. According to Proposition 3.28, there are at most k elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ that map Q to P, so all in all the set S has $O(mn)$ elements.

The sets E and M can both be determined naively in in $O(m^2)$ time and take $O(m)$ memory. The set S has $O(m \cdot n^3)$ elements by Proposition 3.28. We can enumerate orbit representatives of degree k irreducible binary forms in time $\widetilde{O}(q^{k-3})$ and $O(\log q)$ memory using Howe's algorithms. In Step 3(b), we consider $O(q^{n-k})$ values of G. Since m and n are constant for our complexity analysis, the algorithm takes $\widetilde{O}(q^{n-3})$ time and $O(\log q)$ memory. □

The algorithm for Case B is essentially the same as Algorithm 4.14, but now we use orbit representatives for pairs of quadratic factors provided by Howe [How25b, Theorems 3.4 and 3.8], and the lambda orbit invariant as defined in Definition 4.8.

Algorithm 4.16. Enumerating monic m^{th}-power free binary forms of Case B.

Input: A finite field $\mathbb{F}_q$, an integer $m \geq 2$ that divides $q-1$, and a valid square-free factorisation type $D = (d_i)_{1\leq i\leq m-1}$ such that at least one d_i is equal to or larger than 4.

Output: Orbit representatives and their stabilisers for the monic binary forms of $\mathcal{B}_m^D(\mathbb{F}_q)$ of Case B.

1. Determine the group E consisting of the $\overline{e} \in (\mathbb{Z}/m\mathbb{Z})^*$ that fix D.
2. Determine a complete set of representatives M of $\{1, 2, \ldots, m-1\}$ under multiplication by E modulo m.
3. For every $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit representative (P_1, P_2) of pairs of distinct irreducible quadratic binary forms over $\mathbb{F}_q$ given by [How25a, Theorems 3.4 and 3.8], and every $m_P \in M$ such that $d_{m_P} \geq 4$, do:
   (a) Let D' be the square-free factorisation type left after choosing P_1 with multiplicity m_P and P_2 with multiplicity m_P.
   (b) For every monic m^{th}-power free binary form G over $\mathbb{F}_q$ of type D' that satisfies the following:
      (x) P_1 and P_2 do not divide G;
      (y) G has no irreducible factor of degree larger than 2;
      (z) $(P_1P_2)^{m_P} \cdot G$ does not have a pair of irreducible quadratic factors (Q_1, Q_2) of the same multiplicity such that $\lambda(Q_1Q_2) < \lambda(P_1P_2)$;

      do:
      (i) Let $H := (P_1P_2)^{m_P} \cdot G$.
      (ii) Determine the set S of $(\overline{e}, \Gamma) \in E \times \mathrm{PGL}_2(\mathbb{F}_q)$ such that P_1 and P_2 divide $\Gamma(H)^{\overline{e}}$ and their multiplicity is an element of M. Appendix A shows us how to find the Γ that map a pair of irreducible quadratic factors of H with the same lambda invariant as P_1P_2 to P_1P_2.
      (iii) Output H and its stabiliser if $H = \min_{(\overline{e},\Gamma)\in S}\{\Gamma(H)^{\overline{e}}\}$, and continue with the next value of G. The stabiliser of H is contained in S and is found while applying the elements of S to H.

Proposition 4.17. *Algorithm 4.16 is correct and runs in time* $\widetilde{O}(q^{n-3})$ *and* $O(\log q)$ *memory, where* $n = \sum_{i=1}^{m-1} d_i$.

Proof. The proof is analogous to that of Proposition 4.15, apart from the λ orbit invariant taking the roll of the cross polynomial. □

In Case C the roles of the quadratic factors is 'asymmetric', and we encounter a problem when using the λ invariant. Suppose that we are given two distinct multiplicities $m_1, m_2 \in \{1, 2, \ldots, m-1\}$, and that P_1 and P_2 are two distinct monic irreducible quadratic binary forms. We would like to use the $\lambda(P_1P_2)$ orbit invariant for binary forms of the form $P_1^{m_1}P_2^{m_2}$, but $\lambda(P_1P_2)$ is symmetric in P_1 and P_2, while $P_1^{m_1}P_2^{m_2}$ is not. The following lemma shows that $P_1^{m_1}P_2^{m_2}$ and $P_2^{m_1}P_1^{m_2}$ belong to the same $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit, and that the λ invariant thus makes sense for this scenario.

Lemma 4.18. *Let* $G, H \in \mathbb{F}_q[X, Z]$ *be two monic irreducible quadratic binary forms. Then there exists a* $\Gamma \in \mathrm{PGL}_2(\mathbb{F}_q)$ *such that* $\Gamma(G) = H$ *and* $\Gamma(H) = G$.

Proof. If G and H are the same, the identity element of $\mathrm{PGL}_2(\mathbb{F}_q)$ suffices. Suppose that G and H are not equal. Let $\alpha, \beta \in \mathbb{P}^1(\mathbb{F}_{q^2})$ be zeroes of G and H respectively. According to [How25a, Cor. 3.2], there is a unique element of $\mathrm{PGL}_2(\mathbb{F}_q)$ that swaps α with β and α^q with β^q, which thus swaps G and H. □

We start by considering the possible combinations of multiplicities of which we have an irreducible quadratic factor. We encode this as a tuple $(b_i)_i \in \{\texttt{true}, \texttt{false}\}^{m-1}$. Like with the square-free factorisation of a binary form, the action of $(\mathbb{Z}/m\mathbb{Z})^*$ simply permutes this tuple. If there is an element $\overline{e} \in (\mathbb{Z}/m\mathbb{Z})^*$ that fixes our square-free factorisation type, then we should not consider both $(b_i)_i$ and its permutation under $\overline{e}$. Therefore we create a set $B \subset \{\texttt{true}, \texttt{false}\}^{m-1}$ of representatives up to these permutations in Step 2.

Algorithm 4.19. Enumerating monic m^{th}-power free binary forms of Case C.

Input: A finite field $\mathbb{F}_q$, an integer $m \geq 2$ that divides $q-1$, and a valid square-free factorisation type $D = (d_i)_{1 \leq i \leq m-1}$ such that at least two of the d_i are 2 or larger.

Output: Orbit representatives and their stabilisers for the monic binary forms of $\mathcal{B}_m^D(\mathbb{F}_q)$ of Case C.

1. Determine the group E consisting of the $\overline{e} \in (\mathbb{Z}/m\mathbb{Z})^*$ that fix D.
2. Determine a complete set of representatives B of $(b_i)_i \in \{\texttt{true}, \texttt{false}\}^{m-1}$ under permutation by elements of E such that $d_i \geq 2$ whenever b_i is `true`, and at least two of the b_i are `true`.
3. For every $\mathrm{PGL}_2(\mathbb{F}_q)$ orbit representative (P_1, P_2) for pairs of distinct monic irreducible quadratic binary forms given by [How25a, Theorems 3.4 and 3.8], and every $(b_i)_i \in B'$, do:
   (a) Determine the group E' consisting of the $\overline{e} \in E$ that fix $(b_i)_i$ under permutation.
   (b) Let m_1 and m_2 be the first two indices i such that b_i is `true`.
   (c) Let D' be the square-free factorisation type left after choosing P_1 with multiplicity m_1 and P_2 with multiplicity m_2.
   (d) For every monic m^{th}-power free binary form G over $\mathbb{F}_q$ of type D' satisfying the following:
      (w) P_1 and P_2 do not divide G;
      (x) G has no irreducible factor of degree larger than 2;
      (y) $P_1^{m_1} \cdot P_2^{m_2} \cdot G$ has precisely one irreducible quadratic factor of multiplicity i if b_i is `true`, and no irreducible quadratic factors of other multiplicities;
      (z) There exists no $\overline{e} \in E'$ such that $(P_1^{m_1} \cdot P_2^{m_2} \cdot G)^{\overline{e}}$ has two quadratic factors P_3 and P_4 of multiplicities m_1 and m_2 respectively and $\lambda(P_3P_4)$ is smaller than $\lambda(P_1P_2)$;

      do:
      (i) Let $H := P_1^{m_1} \cdot P_2^{m_2} \cdot G$.

(ii) Determine the set S of $(\overline{e}, \Gamma) \in E' \times \mathrm{PGL}_2(\mathbb{F}_q)$ such that P_1 divides $\Gamma(H)^{\overline{e}}$ with multiplicity m_1 and P_2 divides $\Gamma(H)^{\overline{e}}$ with multiplicity m_2. Appendix A shows us how to find the Γ that map a pair of irreducible quadratic factors of H with the same lambda invariant as P_1P_2 to P_1P_2.

(iii) Output H and its stabiliser if $H = \min_{(\overline{e},\Gamma)\in S}\{\Gamma(H)^{\overline{e}}\}$, and continue with the next value of G. The stabiliser of H is contained in S and is found while applying the elements of S to H.

Proposition 4.20. *Algorithm 4.19 is correct and runs in time* $\widetilde{O}(q^{n-3})$ *and* $O(\log q)$ *memory, where* $n = \sum_{i=1}^{m-1} d_i$.

Proof. Let us consider the values that H of Step 3(d)(i) takes. These are precisely the monic binary forms $H \in \mathcal{B}_m^D(\mathbb{F}_q)$ of Case C satisfying the following:

1. H has square-free factorisation type $(d_i)_i$, due to Steps 3(d);
2. The tuple $(b_i)_i \in \{\texttt{true}, \texttt{false}\}^{m-1}$ such that b_i is `true` if and only if H has a quadratic factor of multiplicity i is an element of B, due to Condition 3(d)(y);
3. The quadratic factors P_1 and P_2 of H of the two smallest multiplicities with quadratic factors m_1 and m_2 appear in the representatives considered in Step 3;
4. Amongst the elements of $\{H^{\overline{e}} : e \in E'\}$, there is no element such that the λ invariant of its quadratic factors of multiplicity m_1 and m_2 is smaller than the λ invariant of the quadratic factors of multiplicity m_1 and m_2 of H, due to Condition 3(d)(z).

The only possibility for duplicates of the same orbit now comes from the elements of $\{H^{\overline{e}} : e \in E'\}$ such that these λ invariants match, and by Lemma 4.18, these are precisely the images of H under the elements of S. By construction of B, every orbit of $\mathcal{B}_m^D(\mathbb{F}_q)\big/\mathbb{F}_q^*$ of Case C has an element G that satisfies the first two properties, and we can use the action of $E' \times \mathrm{PGL}_2(\mathbb{F}_q)$ to move G to an element of its orbit that satisfies all properties.

Determining the set S is mostly analogous to constructing the set S of Algorithms 4.14 and 4.16, only this time we keep track of all pairs of quadratic factors (P_3, P_4) with $\lambda(P_3P_4) = \lambda(P_1P_2)$ while checking Condition 3(d)(z). The set B can be determined naively in time $O(2^{m-1})$ and takes $O(2^{m-1})$ memory. The set E' can be determined naively in time $O(2^{m-1}m)$ and takes $O(m)$ memory. The complexity analysis now follows from the same arguments as in Propositions 4.15 and 4.17. □

The final two algorithms are based on Lemma 4.2, which tells us there is only one orbit for square-free binary forms of Galois type $(2, 1)$, and only one orbit for binary forms of Galois type $(1, 1, 1)$.

Algorithm 4.21. Enumerating monic m^{th}-power free binary forms of Case D.

Input: A finite field $\mathbb{F}_q$, an integer $m \geq 2$ that divides $q - 1$, and a valid square-free factorisation type $D = (d_i)_{1\leq i\leq m-1}$ of order $n \geq 3$ such that at least one of the d_i is 2 or larger.

Output: Orbit representatives and their stabilisers for monic binary forms of $\mathcal{B}_m^D(\mathbb{F}_q)$ of Case D.

1. Find an irreducible quadratic binary form P using the methods of [Sho90].
2. Determine the group E consisting of the $\overline{e} \in (\mathbb{Z}/m\mathbb{Z})^*$ that fix D.
3. Determine a complete set of representatives M of $\{1, 2, \ldots, m-1\}$ under multiplication by E modulo m.
4. For every every $m_P \in M$ such that $d_{m_P} \geq 2$, do:
   (a) Determine the group E' consisting of the $\overline{e} \in E$ that fix m_P.
   (b) Let m_Z be the smallest multiplicity such that $P^{m_P} \cdot Z^{m_Z}$ can divide a binary form of square-free factorisation type D.
   (c) Let D' be the square-free factorisation type left after choosing P with multiplicity m_P and Z with multiplicity m_Z.
   (d) For every monic m^{th}-power free binary form G over $\mathbb{F}_q$ of type D' such that Z does not divide G and G splits into linear factors, do:
      (i) Let $H := P^{m_P} \cdot Z^{m_Z} \cdot G$.
      (ii) Let S be the set of $(\overline{e}, \Gamma) \in E' \times \mathrm{PGL}_2(\mathbb{F}_q)$ such that Z divides $\Gamma(H)^{\overline{e}}$ with multiplicity m_Z and $\Gamma(P) = P$. The Γ that fix P and send a given element of $\mathbb{P}^1(\mathbb{F}_q)$ to $(1:0)$ are given in Table 6.
      (iii) Output H and its stabiliser if $H = \min_{(\overline{e},\Gamma)\in S}\{\Gamma(H)^{\overline{e}}\}$, and continue with the next value of G. The stabiliser of H is contained in S and is found while applying the elements of S to H.

Proposition 4.22. *Algorithm 4.21 is correct and runs in time $\widetilde{O}(q^{n-3})$ if $n \geq 4$ and $\widetilde{O}(\sqrt{q})$ if $n = 3$, and takes $O(\log q)$ memory, where $n = \sum_{i=1}^{m-1} d_i$.*

Proof. Let $G \in \mathcal{B}_m^D(\mathbb{F}_q)$ be a monic binary form of Case D. Using the action of $(\mathbb{Z}/m\mathbb{Z})^*$, we can find an element of the orbit of G that has square-free type D. We know that this element has exactly one irreducible quadratic factor Q, and using E we can make sure that the multiplicity of Q is an element of M. According to Lemma 4.2, we can now use the action of $\mathrm{PGL}_2(\mathbb{F}_q)$ to move Q to our fixed quadratic P and one of the linear factors of minimal multiplicity amongst all linear factors to Z.

We now have an element of the orbit of G that will at some point be the value of H in Step 4(d)(iii). The only other elements of the orbit of G that will reach this step are those that also have factor P with multiplicity m_P and factor Z with multiplicity m_Z, and the set S provides all of those elements when applied to H.

Let us consider the set S. For any of the $n-2$ linear factors of H, there are at most 2 elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ that map said factor to Z and stabilise P, which we give explicitly in Table 6. Therefore the set S has $O(mn)$ elements, and each can be determined in time constant in q.

Using the methods described in [Sho90, Section 4], we may construct our single irreducible quadratic binary form in time $\widetilde{O}(\sqrt{q})$. The remainder of the complexity analysis is analogous to that of Proposition 4.15. □

Algorithm 4.23. Enumerating monic m^{th}-power free binary forms of Case E.

Input: A finite field $\mathbb{F}_q$, an integer $m \geq 2$ that divides $q-1$, and a valid square-free factorisation type $(d_i)_{1\leq i\leq m-1}$ such that at least three d_i are non-zero.

Output: Orbit representatives and their stabilisers for monic binary forms of $\mathcal{B}_m^D(\mathbb{F}_q)$ of Case E.

1. Determine the group E consisting of the $\overline{e} \in (\mathbb{Z}/m\mathbb{Z})^*$ that fix D.
2. Choose $m_1, m_2, m_3 \in \{1, 2, \ldots, m-1\}$ such that $Z^{m_1} \cdot X^{m_2} \cdot (X-Z)^{m_3}$ can divide a binary form of square-free factorisation type D.
3. Let D' be the square-free factorisation type left after choosing Z with multiplicity m_1, X with multiplicity m_2, and $X-Z$ with multiplicity m_3.
4. For every monic m^{th}-power free binary form G over $\mathbb{F}_q$ of type D' such that Z, X, and $X-Z$ do not divide G and G splits into linear factors, do:
   a. Let $H := Z^{m_1} \cdot X^{m_2} \cdot (X-Z)^{m_3} \cdot G$.
   b. Determine the set S of $(\overline{e}, \Gamma) \in E \times \mathrm{PGL}_2(\mathbb{F}_q)$ such that Z, X, and $X-Z$ divide $\Gamma(H)^{\overline{e}}$ with multiplicity m_1, m_2, and m_3 respectively. Appendix A shows us how to construct the unique Γ that maps a given triple of linear factors of H to Z, X, and $X-Z$.
   c. Output H and its stabiliser if $H = \min_{(\overline{e},\Gamma)\in S}\{\Gamma(H)^{\overline{e}}\}$, and continue with the next value of G. The stabiliser of H is contained in S and is found while applying the elements of S to H.

Proposition 4.24. *Algorithm 4.23 is correct and runs in time $\widetilde{O}(q^{n-3})$ and $O(\log q)$ memory, where $n = \sum_{i=1}^{m-1} d_i$.*

Proof. Let $G \in \mathcal{B}_m^D(\mathbb{F}_q)$ be a monic binary form of Case E. Using the action of $(\mathbb{Z}/m\mathbb{Z})^*$, we can find an element of the orbit of G that has square-free factorisation type D. According to Lemma 3.26, we can use the action of $\mathrm{PGL}_2(\mathbb{F}_q)$ to move a three linear factors, one each of multiplicity m_1, m_2, and m_3, to Z, X, and $X-Z$ respectively. This element of the orbit of G will at some point be the value of H in Step 4(a), and the other elements of the orbit of G that will be considered at this step, are those obtained by applying elements of S to H.

Determining the set S is very straightforward for this algorithm. Given any ordered triple of linear factors, we can use Table 5 to find the only element of $\mathrm{PGL}_2(\mathbb{F}_q)$ that maps these three factors to Z, X, and $X-Z$ respectively. Therefore the set S has $O(mn^3)$ elements, and we can determine these in time constant in q. The remainder of the complexity analysis is analogous to that of Proposition 4.15. □

Proof of Theorem 4.10. We must find a primitive element of $\mathbb{F}_q$ once for Algorithm 4.12, which can be done in time $O(q^{1/4+\epsilon})$ for any $\epsilon > 0$ using [Shp96]. Propositions 4.15, 4.17, 4.20, 4.22 and 4.24 tell us we can find orbit representatives, along with their stabilisers, of $\mathcal{B}_m^D(\mathbb{F}_q)\big/\mathbb{F}_q^*$ under the action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ in time $\widetilde{O}(q^{n-3})$ if $n \geq 4$ and time $\widetilde{O}(\sqrt{q})$ if $n = 3$, and using $O(\log q)$ memory. Proposition 4.13 shows that applying Algorithm 4.12 to all $O(q^{n-3})$ representatives and stabilisers output by Algorithms 4.14, 4.16, 4.19, 4.21 and 4.23, takes time $\widetilde{O}(q^{n-3})$ if $n \geq 4$ and time $\widetilde{O}(\sqrt{q})$ if $n = 3$, uses $O(1)$ memory, and yields the desired representatives. □

5 Implementation

Our implementation may be found on our public GitHub repository [Lem26]. The function `superelliptic_orbits` defined in `src/superelliptic.jl` implements the algorithms of Section 4.2, and takes 3 arguments:

1. `R::FqPolyRing`, a univariate polynomial ring over a finite field $\mathbb{F}_q$;
2. `m::Integer`, the exponent of the superelliptic function fields to enumerate;
3. `sqfree_type::Tuple{Vararg{Integer}}`, a tuple of integers representing a square-free factorisation type $D = (d_i)_{1 \leq i < m}$.

Before running the algorithms, we check whether the input describes a valid configuration, meaning m divides $q - 1$, and D is a valid square-free factorisation type for $\mathcal{B}_m(\mathbb{F}_q)$, as defined in Section 3.3. The arguments of all other functions are also typed, but further constraints are not checked, since they are not exported to the user, and we assume they are only called with valid input.

We represent a monic m^{th} power free binary form G by the dehomogenisation of its square-free factorisation, by which we mean the tuple $(g_i)_{1 \leq i < m}$ of monic, square-free, pairwise coprime polynomials such that $G = \prod_{i=1}^{m-1} G_i^i$, where G_i is g_i homogenised to degree d_i.

The output of `superelliptic_orbits` consists of orbit representatives for $\mathcal{B}_m^D(\mathbb{F}_q) \big/ \mathbb{F}_q^*$ up to the action of $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ in the aforementioned representation, along with a subset of $\mathbb{F}_q^*$. This set of scalars provides the multiples of the associated monic representative that form the preimages of the surjection of Corollary 3.22 without duplicates, thus enumerating the isomorphism classes of superelliptic function fields over $\mathbb{F}_q$ of exponent m and square-free factorisation type equivalent to D.

We also provide several tests for our implementation, which may be found in the file `test/tests.jl`. In the function `run_tests_super`, we test the algorithms of Section 4.2 separately for inputs that go well together with the case they cover. We also test our implementations of Howe's algorithms for square-free and irreducible binary forms for a variety of degrees and finite fields of different characteristic in the function `run_tests_hyper`. We check the output by applying $(\mathbb{Z}/m\mathbb{Z})^* \times \mathrm{PGL}_2(\mathbb{F}_q)$ (for `run_tests_super`) or $\mathrm{PGL}_2(\mathbb{F}_q)$ (for `run_tests_hyper`) to all representatives in the output, and by checking whether they all have distinct orbits and all orbits together give the expected number of binary forms. This brute-force approach to testing means we can test only relatively small cases.

5.1 Ordering Finite Fields

For our algorithms, we need total orderings on the finite field $\mathbb{F}_q$, the polynomial ring $\mathbb{F}_q[X]$, and the binary forms of $\mathbb{F}_q[X, Z]$. In order to achieve our desired complexity, we need to be able to evaluate these orderings in time polynomial in $\log q$. For the complexity analysis the degrees of the polynomials and binary forms are fixed.

OSCAR does not provide any of these orderings, so we implemented them ourselves. For univariate polynomial rings, we first order them by degree, and for polynomials of the same degree, we implement a lexicographical ordering on

the polynomial coefficients. Since we only ever need to compare binary forms of the same degree, it is sufficient to compare their dehomogenisations instead using this same ordering. As a function of q, this takes $O(1)$ comparisons in $\mathbb{F}_q$, so it suffices to show that we can compare finite field elements efficiently.

For a prime field $\mathbb{F}_p$ for some prime number p, we lift its elements to $\mathbb{Z}$ and compare them in $\mathbb{Z}$. Non-prime finite fields $\mathbb{F}_{p^n}$, where $n > 1$, are represented in OSCAR as a quotient ring $\mathbb{F}_{p^m}[X]/(f(X))$ for some positive integer $m < n$ that divides n, and an irreducible polynomial $f \in \mathbb{F}_{p^m}[X]$ of degree $\frac{n}{m}$. In this case, we lift elements of $\mathbb{F}_{p^n}$ to representatives in $\mathbb{F}_{p^m}[X]$ of degree smaller than $\frac{n}{m} \in O(\log q)$, and use our above ordering on univariate polynomials. If $m \neq 1$, this means we have to repeat this process recursively until we end up in a prime field. All in all, we end up doing $O(\log q)$ comparisons in $\mathbb{Z}$.

5.2 Constructing Binary Forms

In our algorithms, we often need to iterate over all monic binary forms of a given square-free factorisation type D of order n that do not have irreducible factors of a degree larger than some given $s \in \mathbb{Z}_{>0}$. Often, we also need to know which factors of degree s these binary forms contain. See, for example, Step 3(b) of Algorithm 4.14. Importantly, we need to to this using $O(\log q)$ memory, and time $\widetilde{O}(q^n)$, where n is seen as constant for our complexity analysis.

Let us start with iterating over the monic square-free binary forms of degree n and upper bound $s \in \mathbb{Z}_{>0}$ on the degrees of the irreducible factors. We do this by iterating over all $O(q^n)$ monic polynomials f of degree n or $n-1$, which we will later homogenise to degree n, using the lazy iterators of OSCAR. For each of these, we test whether they are sqaure-free, which can be done using a gcd operation on f and its formal derivative and which is implemented in OSCAR. Then, we use OSCAR's distinct-degree factorisation [GG99, Section 14.2], which will tell us whether f contains irreducible factors of degree larger than s. It also gives us the factor f_s of f that is the product of all degree s irreducible factors of f. Now we only need to factor f_s and we will have all the information we need. All of these steps only take time polynomial in $\log q$ per binary form [GG99, Theorem 14.4].

An alternative method for iterating over these square-free binary forms, is to iterate over all Galois types they could have, and constructing all binary forms of those Galois types by building them out of irreducible factors. However, this means that we would have to iterate over the irreducible polynomials of degree less than or equal to s several times, since we can not store these polynomials in $O(\log q)$ memory. If we were to store these polynomials, we could reuse them throughout our program, and we would not have to factor any binary forms. For small values of q, this trade-off between memory and efficiency is worth considering.

In order to iterate over all monic binary forms of a given square-free factorisation type $(d_i)_i$ with an upper bound s on the degrees of the irreducible factors, we construct square-free factorisations by taking combinations of square-free binary forms of each degree d_i with the same restriction on irreducible factors, and by using gcd calculations to determine whether they are coprime.

6 Results

In this section, we will provide timings for our algorithm on several inputs, and we will summarise the output of Julia's sampling profiler, as described in Section 2.3. We discuss the results in Section 7.

6.1 Benchmarks

All benchmarks and their configurations may be found in `test/benchmarks.jl` on our GitHub repository [Lem26]. We aim to run every case 10 times in order to provide good averages, but we stopped early if the previous runs have already taken over half an hour. Our results were obtained using Julia 1.12.6 and OSCAR 1.7.3 on a system with an AMD Ryzen 5 9600X processor and 24GB of RAM.

We will start by enumerating superelliptic function fields of exponent 4 and square-free factorisation type $(2, 4, 2)$. This input was chosen because there are relatively many small fields $\mathbb{F}_q$ such that $4 \mid q-1$, and because the type $(2, 4, 2)$ has a non-trivial $(\mathbb{Z}/4\mathbb{Z})^*$ stabiliser and covers all cases of our algorithm as described in Section 4.2.

Mimicking Howe, we also report the quantity $10^5 \cdot t/q^5$, where t is the time taken in seconds and q is the field size. Since our algorithm has time complexity $\widetilde{O}(q^5)$, this quantity should grow at most polynomial in $\log q$. The results may be found in Table 2.

Field size	Time (s)	$10^5 \cdot t/q^5$
5	0.35	11.15
9	29.60	50.13
13	106.67	28.73
17	406.72	28.64

Table 2: Average runtime in seconds over 10 runs for enumerating superelliptic function fields of exponent 4 and square-free factorisation type $(2, 4, 2)$.

In Table 2, the quantity $10^5 \cdot t/q^5$ is perhaps not as consistent as we would like it to be. However, we note that $\mathbb{F}_5$ only has 6 linear binary forms, and therefore no binary forms of square-free factorisation type $(2, 4, 2)$ of Cases D and E. The relatively poor performance in $\mathbb{F}_9$ might be explained by the fact that this is the only non-prime field we tested.

We will continue by comparing the performance of Howe's implementation [How25c] in the programming language Magma [BCP97] with our own implementation of his algorithm for hyperelliptic function fields of square-free factorisation type (6), which he refers to as hyperelliptic curves of genus 2. His results were obtained using Magma V2.28-20 on a system with an Apple M4 Max processor and 64GB of RAM. This time, we report the quantity $10^5 \cdot t/q^3$, because the time complexity should be $\widetilde{O}(q^3)$. The results may be found in Table 3.

Field size	Our implementation			Howe's implementation	
	Time (s)	Runs	$10^5 \cdot t/q^3$	Time (s)	$10^5 \cdot t/q^3$
23	15.45	10	126.79	-	-
31	41.33	10	138.72	0.92	3.09
43	80.27	10	100.96	-	-
61	232.92	8	102.62	5.11	2.25
89	732.13	3	103.85	-	-
127	2197.08	1	107.26	34.72	1.69

Table 3: Average runtime in seconds for enumerating hyperelliptic function fields of square-free factorisation type (6). The results of Howe's implementation were taken from [How25b, Table 2].

Finally, we ran our algorithm for a variety of exponents and square-free factorisation types while fixing the field size at $q = 2401$. The results may be found in Table 4.

Exponent	Square-free type	Time (ms)
3	$(2, 2)$	2768
4	$(1, 2, 1)$	255
6	$(0, 1, 2, 1, 0)$	270
12	$(0, 0, 0, 1, 0, 1, 2, 0, 0, 0, 0)$	379

Table 4: Average runtime in milliseconds over 10 runs for enumerating superelliptic function fields over $\mathbb{F}_{7^4}$ of varying exponents and types taken from [Moo10].

Remark 6.1. The cases considered in Table 4 are cases (3), (4), (5), and (20) from [Moo10, Table 1], in which Ben Moonen proves that these are 4 of the 20 possible configurations that give rise to special subvarieties of the moduli space of Abelian varieties.

6.2 Profiling

We ran Julia's sampling profiler [Jule] while enumerating superelliptic function fields over $\mathbb{F}_{13}$ of exponent 4 and square-free factorisation type $(2, 4, 2)$. Of our algorithms of Section 4.2, 30% of the samples were collected in Algorithm 4.14, 32% was collected in Algorithm 4.16, 11% in Algorithm 4.19, 3% in Algorithm 4.21, 1% in Algorithm 4.23, and 3% in the algorithm of Section 4.2.1.

The remaining 20% is difficult to attribute, but consists at least in part of Howe's algorithms for enumerating irreducible binary forms of degree 4 and pairs of irreducible quadratic binary forms, and the runtime overhead of lazy iteration and just-in-time compilation.

We further examined the function calls made by Algorithms 4.14 and 4.16. Both spend over half of their runtime on finding the elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ that maps a given (pair of) irreducible binary form(s) to another, as described in Appendix A. The most costly part of this process seems to be finding roots of irreducible binary forms using the `roots` function provided by the FLINT C library [FLI26]. Algorithms 4.14 and 4.16 spend most of the remainder of their runtime on applying elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ to binary forms.

7 Discussion and Future Work

In Section 4.2, we succeeded in providing an algorithm that enumerates superelliptic function fields over finite fields, of a given exponent and square-free factorisation type, in quasilinear time in the size of the output, and logarithmic memory complexity. As we can see in Table 4, our implementation of this algorithm performs quite well, even in $\mathbb{F}_{2401}$, on several square-free factorisation types that are interesting despite being of order 4. For square-free factorisation types of larger order, such as in Table 2, we can see that the algorithm struggles with larger fields, which is to be expected from our complexity analysis.

Our algorithms can also enumerate hyperelliptic function fields. In Table 3, we compared our implementation to Howe's implementation, but we should emphasise that these results were obtained on different machines in different programming languages, and are thus not directly comparable. Additionally, Howe only implements his algorithm for square-free binary forms of degree 6, 8, and 10, and he regularly deviates from the algorithm as described in his paper to handle cases specific to these degrees in a more efficient way. These optimisations are not trivial to generalise to larger degrees or even the superelliptic case, and in future work we could investigate to what extent this is possible.

The main advantage of our implementation is that it works for a much larger variety of function fields. Not only can we enumerate the hyperelliptic function fields for which Howe did not implement his algorithm, but we can also enumerate superelliptic function fields. Additionally, Julia and OSCAR are open source, whereas Magma is proprietary, making our implementation more accessible.

Another advantage of our implementation is the lazy iteration style of returning results. Since our time complexity is quasilinear in the size of the output, the average time to find the next result should only grow polynomial in $\log q$. This is especially useful if you do not need a complete list of representatives. For example, if you are trying to find a single superelliptic function field with a given property that is invariant under isomorphism, you could continue to request representatives only until you find one that fits your requirements. Due to the lack of support for this kind of iteration in Magma, Howe only emulates lazy iteration by writing results to files in secondary memory.

In future work, we could attempt to find more efficient ways to determine the elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ that map a given irreducible binary form to another of the same degree, because we saw in Section 6.2 that our current method, as described in Appendix A, was a major bottleneck of our algorithm. We could also consider to represent these irreducible binary forms of degree d by their roots in a fixed copy of $\mathbb{F}_{q^d}$, as these are needed by our current method and can be, as our profiling showed, expensive to find.

References

[Aud02] Michèle Audin. *Geometry*. 1st ed. Springer Berlin, Heidelberg, Sept. 2002. DOI: 10.1007/978-3-642-56127-6.

[BCP97] Wieb Bosma, John Cannon and Catherine Playoust. 'The Magma algebra system. I. The user language'. In: *J. Symbolic Comput.* 24.3-4 (1997). Computational algebra and number theory (London, 1993), pp. 235–265. ISSN: 0747-7171. DOI: 10.1006/jsco.1996.0125.

[Bez+17] Jeff Bezanson et al. 'Julia: A fresh approach to numerical computing'. In: *SIAM review* 59.1 (2017), pp. 65–98. DOI: 10.1137/141000671. URL: https://julialang.org/.

[Fie+17] Claus Fieker et al. 'Nemo/Hecke: Computer Algebra and Number Theory Packages for the Julia Programming Language'. In: *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*. ISSAC '17. New York, NY, USA: ACM, 2017, pp. 157–164. DOI: 10.1145/3087604.3087611.

[FLI26] The FLINT team. *FLINT: Fast Library for Number Theory*. Version 3.5.0. 2026. URL: https://flintlib.org.

[GG99] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. 1st ed. Cambridge University Press, 1999.

[How25a] Everett W. Howe. 'Enumerating hyperelliptic curves over finite fields in quasilinear time'. In: *Research in Number Theory* 11.1 (Jan. 2025), p. 26. ISSN: 2363-9555. DOI: 10.1007/s40993-024-00594-7.

[How25b] Everett W. Howe. *Enumerating places of $\mathbf{P}^1$ up to automorphisms of $\mathbf{P}^1$ in quasilinear time*. 2025. arXiv: 2407.05534 [math.NT].

[How25c] Everett W. Howe. *everetthowe/hyperelliptic*. GitHub. July 2025. URL: https://github.com/everetthowe/hyperelliptic/ (visited on 21/06/2026).

[Jula] The Julia Project. *Benchmarks*. URL: https://julialang.github.io/Microbenchmarks/dev/benchmarks/ (visited on 19/06/2026).

[Julb] The Julia Project. *BenchmarkTools*. URL: https://juliaci.github.io/BenchmarkTools.jl/dev/manual/ (visited on 20/06/2026).

[Julc] The Julia Project. *Iteration Utilities*. URL: https://docs.julialang.org/en/v1/base/iterators/ (visited on 19/06/2026).

[Juld] The Julia Project. *JIT Design and Implementation*. URL: https://docs.julialang.org/en/v1/devdocs/jit/ (visited on 19/06/2026).

[Jule] The Julia Project. *Profiling*. URL: https://docs.julialang.org/en/v1/manual/profile/ (visited on 20/06/2026).

[Knu76] Donald E. Knuth. 'Big Omicron and big Omega and big Theta'. In: *SIGACT News* 8.2 (Apr. 1976), pp. 18–24. ISSN: 0163-5700. DOI: 10.1145/1008328.1008329.

[LA04] Chris Lattner and Vikram Adve. 'LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation'. In: *Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04)*. Palo Alto, California, Mar. 2004. DOI: 10.1109/CGO.2004.1281665.

[Lan12] Serge Lang. *Algebra*. 3rd ed. Graduate Texts in Mathematics. Springer New York, NY, Dec. 2012. DOI: 10.1007/978-1-4613-0041-0.

[Lem26] Rogier Lemaire. *EnumSuperelliptic.jl*. GitHub. June 2026. URL: https://github.com/rogierlemaire/EnumSuperelliptic.jl (visited on 26/06/2026).

[Moo10] Ben Moonen. *Special subvarieties arising from families of cyclic covers of the projective line*. Berlin, Germany, 2010. DOI: 10.4171/DM/314.

[OSC] OSCAR Documentation. *Rational Function Fields*. URL: https://docs.oscar-system.org/stable/AbstractAlgebra/function_field/ (visited on 19/06/2026).

[OSC26] The OSCAR Team. *OSCAR – Open Source Computer Algebra Research system, Version 1.7.3*. 2026. URL: https://www.oscar-system.org/about.

[Rom05] Steven Roman. *Field Theory*. 2nd ed. Graduate Texts in Mathematics. Springer New York, NY, July 2005. DOI: 10.1007/0-387-27678-5.

[Sho90] Victor Shoup. 'New algorithms for finding irreducible polynomials over finite fields'. In: *Mathematics of Computation* 54 (1990), pp. 435–447. DOI: 10.1090/S0025-5718-1990-0993933-0.

[Shp96] Igor Shparlinski. 'On finding primitive roots in finite fields'. In: *Theoretical Computer Science* 157.2 (1996), pp. 273–275. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(95)00164-6.

[Ste22] Peter Stevenhagen. *Algebra II*. Universiteit Leiden, Leiden, Aug. 2022. URL: https://websites.math.leidenuniv.nl/algebra/.

[Ste25] Peter Stevenhagen. *Algebra III*. Universiteit Leiden, Leiden, Jan. 2025. URL: https://websites.math.leidenuniv.nl/algebra/.

[Sti08] Henning Stichtenoth. *Algebraic Function Fields and Codes*. 2nd ed. Graduate Texts in Mathematics. Springer Berlin, Heidelberg, Nov. 2008. DOI: 10.1007/978-3-540-76878-4.

A Constructing Elements of the Projective Linear Group

In the majority of the algorithms of Section 4, we need to be able to find the elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ that map one binary form onto a binary form, for several kinds of binary forms that appear in our algorithms.

Lemma 3.26 tells us that there is precisely one element of $\mathrm{PGL}_2(K)$ that maps a given distinct triple of elements of $\mathbb{P}^1(K)$ to another such given triple. Let us discuss how to construct this element of $\mathrm{PGL}_2(K)$. Let $a, b, c \in K$ be distinct, and consider the following four elements of $\mathrm{PGL}_2(K)$:

$\Gamma \in \mathrm{PGL}_2(K)$	$\Gamma(1:0)$	$\Gamma(0:1)$	$\Gamma(1:1)$
$\left(\begin{smallmatrix} a(c-b) & b(a-c) \\ c-b & a-c \end{smallmatrix}\right)$	$(a:1)$	$(b:1)$	$(c:1)$
$\left(\begin{smallmatrix} c-b & b \\ 0 & 1 \end{smallmatrix}\right)$	$(1:0)$	$(b:1)$	$(c:1)$
$\left(\begin{smallmatrix} a & c-a \\ 1 & 0 \end{smallmatrix}\right)$	$(a:1)$	$(1:0)$	$(c:1)$
$\left(\begin{smallmatrix} a & -b \\ 1 & -1 \end{smallmatrix}\right)$	$(a:1)$	$(b:1)$	$(1:0)$

Table 5: The elements of $\mathrm{PGL}_2(K)$ that we use to construct other elements of $\mathrm{PGL}_2(K)$. The last three columns indicate where they map $(1:0)$, $(0:1)$, and $(1:1)$ respectively.

Using the elements of $\mathrm{PGL}_2(K)$ given in Table 5 and their inverses, we can map a given distinct triple of elements of $\mathbb{P}^1(K)$ to $(1:0)$, $(0:1)$, and $(1:1)$, and then map these to our second distinct triple. We will use this to construct the element of $\mathrm{PGL}_2\left(\overline{\mathbb{F}_q}\right)$ that sends three distinct zeroes of a given binary form to three distinct zeroes of another given binary form.

Let us now discuss how to determine if a given $\Gamma \in \mathrm{PGL}_2\left(\overline{\mathbb{F}_q}\right)$ is an element of $\mathrm{PGL}_2(\mathbb{F}_q)$. Suppose that Γ is represented by $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{GL}_2\left(\overline{\mathbb{F}_q}\right)$. Then Γ is an element of $\mathrm{PGL}_2(\mathbb{F}_q)$ if and only if we can scale M so its coefficients are in $\mathbb{F}_q$. This is the same as saying that the ratios between the non-zero coefficients of M are elements of $\mathbb{F}_q$. Therefore it suffices to scale M such that one of its non-zero coefficients becomes 1, and to check if the other coefficients are elements of $\mathbb{F}_q$, which may be done using the Frobenius map. Using repeated squaring [GG99, Algorithm 4.3], this takes $O(\log q)$ multiplications in $\mathbb{F}_q^*$.

Proposition A.1. *Let Γ be an element of $\mathrm{PGL}_2(\mathbb{F}_q)$, let $(\alpha:1)$ be an element of $\mathbb{P}^1(\overline{\mathbb{F}_q})$, and let $(\beta:1)$ be $\Gamma(\alpha:1)$. Then $\Gamma(\alpha^q:1) = (\beta^q:1)$.*

Proof. Let $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{GL}_2(\mathbb{F}_q)$ be a representative of Γ. Then

$$\Gamma(\alpha^q:1) = (a\alpha^q + b : c\alpha^q + d) = ((a\alpha+b)^q : (c\alpha+d)^q) = (\beta^q:1).$$

□

Corollary A.2. *Let $P, Q \in \mathbb{F}_q[X,Z]$ be two irreducible binary forms of degree $k \geq 3$. Then the set $\{\Gamma \in \mathrm{PGL}_2(\mathbb{F}_q) : \Gamma(P) \equiv Q \pmod{\mathbb{F}_q^*}\}$ contains at most k elements.*

Proof. Such a $\Gamma \in \mathrm{PGL}_2(\mathbb{F}_q)$ must map a given zero $(\alpha:1)$ of P to a zero of Q, and we have k choices for a zero of Q. There can be at most one element of

$\mathrm{PGL}_2(\mathbb{F}_q)$ with this chosen image of $(\alpha : 1)$, because Proposition A.1 tells us that this determines the images of two distinct conjugate zeroes $(\alpha^{q^i} : 1)$ of P, and Lemma 3.26 then gives us the unique element of $\mathrm{PGL}_2\left(\overline{\mathbb{F}_q}\right)$ that maps these three zeroes of P to the specified images. □

For Algorithm 4.14, we need to find the $\Gamma \in \mathrm{PGL}_2(\mathbb{F}_q)$ that map Q to P, where P and Q are irreducible binary forms of the same degree $k \geq 3$. Corollary A.2 shows us which k elements of $\mathrm{PGL}_2\left(\mathbb{F}_{q^k}\right)$ we must consider, and we can use the above approach to check whether they belong to $\mathrm{PGL}_2(\mathbb{F}_q)$.

For Algorithms 4.16 and 4.19, we need to find the $\Gamma \in \mathrm{PGL}_2(\mathbb{F}_q)$ that map Q_1 to P_1 and Q_2 to P_2, where P_1, P_2, Q_1 and Q_2 are irreducible quadratic binary forms such that $P_1 \neq P_2$ and $Q_1 \neq Q_2$. Given one zero each of Q_1 and Q_2, we have two choices for their image: either zero of P_1 and P_2 respectively. The images of the other zeroes of Q_1 and Q_2 are then the remaining zeroes of P_1 and P_2. Now we have four elements of $\mathrm{PGL}_2\left(\overline{\mathbb{F}_q}\right)$ to construct using Table 5, after which we check if they belong to $\mathrm{PGL}_2(\mathbb{F}_q)$.

Finally, for Algorithm 4.21, we need to find the $\Gamma \in \mathrm{PGL}_2(\mathbb{F}_q)$ that fix a given irreducible quadratic binary form P, and map a given $(a : b) \in \mathbb{P}^1(\mathbb{F}_q)$ to $(1 : 0) \in \mathbb{P}^1(\mathbb{F}_q)$. We only have two options in this case: either we swap the zeroes of P, or we leave them both in place. Let $\alpha \in \mathbb{F}_{q^2}$ be a zero of $P(X, 1)$, and let n and t be the norm and trace of α over $\mathbb{F}_q$ respectively. One may check that the following are the desired elements of $\mathrm{PGL}_2(\mathbb{F}_q)$:

	$\Gamma(\alpha) = \alpha$	$\Gamma(\alpha) = \alpha^q$
$(a : b) = (1 : 0)$	$\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} -1 & t \\ 0 & 1 \end{smallmatrix}\right)$
$(a : b) = (c : 1)$	$\left(\begin{smallmatrix} c-t & n \\ -1 & c \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} c & n-ct \\ 1 & -c \end{smallmatrix}\right)$

Table 6: The elements of $\mathrm{PGL}_2(\mathbb{F}_q)$ that fix the quadratic minimal polynomial of α and map $(a : b) \in \mathbb{P}^1(\mathbb{F}_q)$ to $(1 : 0)$. The first row gives these matrices if $b = 0$, and the second row gives them if $b \neq 0$.