2023-12-13

Solving norm equations over global function fields using compact representations

Leem, Sumin

UNIVERSITY OF CALGARY

Solving norm equations over global function fields using compact representations

by

Sumin Leem

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN MATHEMATICS AND STATISTICS

CALGARY, ALBERTA

DECEMBER, 2023

Abstract

This thesis provides contributions to improving the efficiency of solving norm equations over global function fields F and providing theoretical and empirical evidence of the improvement. We present two new algorithms for solving norm equations over F; one is an exhaustive search algorithm inspired by the sole existing method in [21], and the other is an algorithm using principal ideal testing via index calculus. We employed compact representations in both of the novel algorithms to represent solutions in less space and also to improve their efficiency. In order to compare the asymptotic and practical efficiency of the novel algorithms to the existing one, we performed rigorous complexity analyses and empirical analyses of the algorithms. We provided the asymptotic complexities of the algorithm as the number of bit operations required. We also analyzed for the first time the complexity of the algorithm for computing compact representations in [15] as a part of the complexity analyses of the novel algorithms. The asymptotic complexities of both of our novel algorithms were reduced by double exponential factors in the size of F, when compared to the existing algorithm in [21]. The new algorithm with principal ideal testing requires even fewer bit operations than the new exhaustive search algorithm in terms of the size of F. Empirical testing results demonstrate our novel algorithms are significantly faster than the existing one in practice as expected from the complexity analyses.

Preface

This thesis is an original work by the author. No part of this thesis has been previously published.

Acknowledgements

I would like to thank my supervisors Dr. Renate Scheidler and Dr. Michael Jacobson for their patience and guidance. I greatly appreciate all their insightful advice and constant support. I would also like to thank Dr. Mark Bauer, Dr. Matthew Greenberg for being my committee members, and Dr. Eva Goedhart and Dr. Kristine Bauer for serving as examiners.

I express my gratitude to Dr. Ha Tran and Dr. Jens Bauch for sharing their number theory knowledge and also being good friends. All the discussions that we had about global fields helped me better understand the topics. My appreciation also goes to the number nosh folks, especially Evan McNeil, Randy Yee, and David Marquis for the meaningful discussions on mathematics and computer science.

I would like to thank all my friends for their support and friendship. Especially, I want to express my sincere gratitude to Jun for his love and encouragement.

Last but not least, my special thanks go to my parents, sister and brother for their love and support throughout these years despite the distance.

Table of Contents

List of Figures

List of Tables

List of Symbols and abbreviations

Symbol	Definition
$\mathbb{Z}$	The ring of integers
$\mathbb{Q}$	The field of rational numbers
$\mathbb{R}$	The field of real numbers
$\mathbb{F}_q$	The field with q elements
$[F' : F]$	The degree of an algebraic extension F'/F
$\lvert S\rvert$	The number of elements in a set S
$\lvert s\rvert$	The absolute value of $s \in \mathbb{R}$
$\lceil s \rceil$	The smallest integer that is greater than or equal to $s \in \mathbb{R}$
$\lfloor s \rceil$	The nearest integer to $s \in \mathbb{R}$
$\lVert v\rVert_\infty$	The maximum norm of a real vector v
ω	The matrix multiplication exponent
HNF	Hermite Normal Form
M	A matrix of real or polynomial entries
$(M)_i$	The i-th row of M
$(M)_{\cdot,j}$	The j-th column of M
$(M)_{i,j}$	The i, j-th entry of M
v, V	A vector of real or polynomial entries
$(v)_i, (V)_i$	The i-th entry of v, The i-th entry of V

Algebraic Function Fields

Symbol	Definition
k	A finite field
$k[x]$	The polynomial ring over a field k in x
$k(x)$	The rational function field over k in x
F/k	An algebraic function field over a field k
n	The extension degree
g	The genus of g
$\mathcal{B}$	A basis of F/k
k_0	The full constant field of F/k
$\mathbf{h}(\lambda)$	The height of $\lambda \in k[x]$
$\mathbf{h}_{\mathcal{B}}(\alpha)$	The height of $\alpha \in F$ with respect to a basis $\mathcal{B}$
O_F	The finite maximal order
$O_{F,\infty}$	The infinite maximal order

$O_F^\times$	The unit group of O_F
$\{\varepsilon_i \mid 1 \leq i \leq r\}$	A system of fundamental units of O_F
r	The unit rank of O_F
Λ	The unit lattice of O_F
R_F	The regulator of O_F
$\mathrm{Norm}_{F/k(x)}(\alpha)$	The norm of $\alpha \in F$
$\mathbb{P}(F)$	The set of all places of F
$\mathbb{P}_0(F)$	The set of all finite places of F
$\mathbb{P}_\infty(F)$	The set of all infinite places of F
$v_P(\alpha)$	The value of α at a place P of F
P	a place in F
$\mathfrak{p}$	A prime ideal of O_F
I	A fractional ideal of O_F
$\mathrm{Id}(O_F)$	The set of nonzero fractional ideals of O_F
$\mathrm{PrincId}(O_F)$	The set of nonzero principal ideals of O_F
$Cl(O_F)$	The ideal class group of O_F
h_{O_F}	The ideal class number of O_F
$\mathrm{Norm}(I)$	The norm of an ideal I of O_F
$\Delta(I)$	The discriminant of I
D	A divisor of F
$\mathrm{Div}(F)$	The divisor group of F
$\mathrm{Div}^0(F)$	The set of all divisors of F of degree 0
$\mathrm{PrincDiv}(F)$	The set of all principal divisors of F
$\mathbf{h}(D)$	The height of D
$L(D)$	The Riemann-Roch space associated to D
$Cl(F)$	The divisor class group of F
h_F	The divisor class number of F
$\|\alpha\|_\infty$	The maximum norm of $\alpha \in F$
$val_\infty(\alpha)$	The vector of the values of $\alpha \in F$ at the infinite places of F
S	A finite set of places of F
O_S	The ring of S-integers
$O_S^\times$	The S-unit group of for a finite set S of places of F
$\{\epsilon_1, \ldots, \epsilon_{\lvert S\rvert-1}\}$	A system of fundamental S-units
r_S	The S-unit rank of O_S
Λ_S	The S-unit lattice of F
Λ'_S	The S-unit valuation lattice of F
R_S	The S-regulator
R'_S	The covolume of Λ'_S
$\mathsf{RR}(h_D)$	The cost of computing a Riemann-Roch space of a divisor D of bounded height $\mathbf{h}(D) \leq h_D$

Chapter 1

Introduction

Solving Diophantine equations over global fields has been a central problem in number theory. As Diophantine equations, norm equations over global fields have many applications in number theory, such as computing the ideal class group of a global field. In this thesis, we will present new and existing algorithms for solving norm equations along with examples and detailed complexity analyses. We use algebraic properties and incorporate compact representations to represent solutions to improve the time and space efficiency of the existing method in [21].

In this introduction, we first provide a brief history of solving norm equations over global fields and using compact representations in global fields in Section 1.1. Then we summarize the contributions of this thesis in Section 1.2. Finally, we give an outline of this thesis in Section 1.3

1.1 Background and motivation

A Diophantine equation is a polynomial equation in at least two unknowns with integer coefficients. Diophantine equations have a long history as an object of widespread interest in mathematics, which originates back to the 3rd century. Solving Diophantine equations is a difficult problem and often requires advanced mathematical theories, even for relatively

simple equations. There have been numerous problems directly related to solving various Diophantine equations. For example, Fermat's last theorem involves a famous Diophantine equation, $x^n + y^n = z^n$. The equation is simple to state, but its general solutions were not found for more than 350 years until Wiles proved that there are no positive integers x, y, and z that satisfy the equation when $n > 2$ by using modern geometry [48]. In addition, Catalan's conjecture is that for $a, b > 1$ and $x, y > 0$, $x^a - y^b = 1$ has the only solution $a = 2$, $b = 3$, $x = 3$, and $y = 2$. It took over 150 years to prove the conjecture is true by advanced theories in algebraic number theory [32].

In number theory, it is natural to consider global fields to generalize Diophantine equations over other domains. There are two kinds of global fields: number fields and global function fields.

A number field K is a field that is a finite extension of the rationals $\mathbb{Q}$. The integral closure of the integers $\mathbb{Z}$ in K is called the ring of integers of K, and denoted by O_K. The ring of integer O_K is a Dedekind domain. A Diophantine equation over a number field is defined as a polynomial equation over O_K in at least 2 unknowns. In number fields K, a special type of Diophantine equation can be defined, which is called a norm equation. The norm of an element $\alpha \in K$ over $\mathbb{Q}$ is well-defined as $K/\mathbb{Q}$ is a finite extension. Denoting the norm of $\alpha \in K$ over $\mathbb{Q}$ by $\mathrm{Norm}_{K/\mathbb{Q}}(\alpha)$, norm equations over number fields are equations of the form $\mathrm{Norm}_{K/\mathbb{Q}}(\alpha) = c$, for some $c \in \mathbb{Q}$. Solving a norm equation over a number field K means, given $c \in \mathbb{Z}$, finding all elements α in O_K that satisfy the equation up to units of O_K. Solving norm equations over K can be used to test the equivalence of two ideals of O_K, and compute the ideal class group of O_K [33, p.379].

Solving norm equations over number fields has been well-studied as a classical problem in number theory. In 1973, Siegel suggested an algorithm for number fields that are Galois over $\mathbb{Q}$. Siegel's algorithm solves norm equations by bounding the absolute values of the solutions [39]. In 1989, Pohst and Zassenhaus introduced a way to solve norm equations over algebraic number fields by an exhaustive search method [33] using inequalities given in [31]. In [16],

Fieker, Jurk, and Pohst provided an algorithm for solving relative norm equations. This algorithm is also an exhaustive search using an enumeration algorithm modified from the one in [17]. Then Simon developed a way to solve norm equations algebraically using S-units in 2002 [41].

A global function field F is an analogue of a number field where the integers and the rationals are replaced by the polynomial ring and the field of rational functions over a finite field, respectively. To be specific, let k be a finite field, and x be transcendental over k. Denote the polynomial ring and the field of rational functions in x over k by $k[x]$ and $k(x)$, respectively. A global function field F/k is an extension field F such that F is a finite algebraic extension of $k(x)$. The integral closure of $k[x]$ in F is called the finite maximal order of F, and denoted by O_F. The finite maximal order O_F is also a Dedekind domain. A Diophantine equation over a global function field is defined the same way as it is over a number field, which is a polynomial equation over O_F in at least two unknowns. As $F/k(x)$ is a finite extension, norm equations can be also defined over global function fields as

$$\mathrm{Norm}_{F/k(x)}(\alpha) = c \tag{1.1}$$

for some rational function $c \in k[x]$. Solving a norm equation over a function field F means finding all elements in O_F that satisfy the equation up to units of O_F.

Norm equations over global function fields are connected to various other number theoretical problems. Similar to number fields, solving norm equations over F can also be used to test the equivalence of two ideals of O_F, and compute the ideal class group of O_F. Moreover, finding solutions of (1.1) in submodules of O_F of rank m when $m = 2$ can be reduced to solving Thue equations [20, 37]. When $2 < m < n$, norm equations can be reduced to S-unit equations [22].

In contrast to the body of research dedicated to solving norm equations over number fields, solving norm equations over function fields is significantly less studied. In function

fields, Gaál and Pohst adopted the exhaustive search method of [33] to the setting of algebraic function fields in 2009 [21].

Unfortunately, the method, as described in [21], is difficult to apply in practice since it does not include an explicit search space. An implementation and a complexity analysis for it are also not available.

In solving norm equations over a global field, a number field or a global function field, and presenting solutions of them, it is required to represent elements of the global field. Denoting a global field by F, there are several ways to represent elements of F. One representation is with respect to an integral basis. Let $\mathcal{B} = \{\omega_i | 1 \leq i \leq n\}$ be an integral basis of F, where n is the extension degree of $F/\mathbb{Q}$ or $F/k(x)$. Then an element $\alpha \in F$ can be written with respect to $\mathcal{B}$

$$\alpha = \sum_{i=1}^{n} \lambda_i \omega_i, \tag{1.2}$$

where $\lambda_i \in k(x)$ for $1 \leq i \leq n$. This representation is commonly used, but it tends to require a huge number of bits to store even when the norm of α is small and $\mathcal{B}$ is a reduced basis which is suitable for computations. For example, the fundamental unit η_F of a real quadratic function field $F_2 = \mathbb{F}_q(x)(\sqrt{D})$ with an odd prime power q and a square-free polynomial $D \in \mathbb{F}_q[x]$ has norm in $\mathbb{F}_q^*$, but the coefficients of η_F with respect to the integral basis $\{1, \sqrt{D}\}$ can have degrees as large as $q^{\deg(D)/2}$, which is exponential in $\deg(D)$. An example of a fundamental unit of a global function field in standard representation can be found in Appendix A because it is too long to provide here. Thus, having a space efficient representation is desirable for computational purposes.

A compact representation is a shorter representation in which an element is presented in power product form. A compact representation of α is stored as

$$\mathbf{t}_\alpha = (\mu, \beta_1, \beta_2, \cdots, \beta_l),$$

where μ and β_i are elements of F specifying a power product with integer exponents ν_i,

typically $\nu_i = 2^{l-i}$ for $i \in \{1, 2, \cdots l\}$,

$$\alpha = \mu \prod_{i=1}^{l} (\beta_i)^{\nu_i},$$

and the size of the representations of μ and β_i with respect to a reduced basis is polynomial in the size of F and of the norm of α. An example of a fundamental unit of a global function field in compact representation can also be found in Appendix A.

In number fields, the idea of using a compact representation for integral elements originated in the early 1990s. Fung and Williams represented fundamental units of complex cubic number fields in a power product form in 1991 [19]. Buchmann, Thiel, and Williams adjusted the power product form to represent quadratic integers in 1995 [11], building on Cohen's unit representation which appeared in [12]. Furthermore, Thiel generalized the representation to elements of number fields of arbitrary degree [46], and proved that the problem of computing the class number of a number field belongs to NP with the use of the compact representation of a system of fundamental units [45]. In 2002, Jacobson and Williams applied this representation to find the coefficients of quadratic field elements with small norms, modulo a positive integer rapidly. These techniques were also used to solve certain Diophantine equations, such as Pell equations [27], and $y^2 = 1^k + 2^k + \cdots + x^k$ for even k [26]. Jacobson and Williams presented the compact representation without logarithm approximation in 2009 [28]. In 2012, Silvester, Jacobson, and Williams adjusted the representation of real quadratic integers to make a shorter representation by reducing the number of terms [40]. They used powers of 3 as exponents instead of powers of 2, so the individual terms are bigger, but the number of terms of the compact representation and the storage of the representation became smaller. They also provided numerical results regarding the sizes of their compact representations.

In global function fields, compact representations are much less studied compared to number fields. The concept of compact representations was introduced for the first time

in 1996 in [35]. There, Scheidler described the compact representation in real quadratic function fields. The representation was used to determine the principality of an integral ideal, equivalence of two ideals, and if one ideal is a power of another ideal. In 2000, Scheidler described further applications showing that various decision problems related to principal ideals, class number, and ideal classes in quadratic function fields with high genus belong to NP using the compact representation [36]. In 2013, a generalized compact representation for global function field elements appeared in Eisenträger and Hallgren [15] with applications, including a proof that the principal ideal problem in global function fields belongs to NP. They also gave polynomial time quantum algorithms for computing a generator of a principal ideal, the unit group, and the class group of a function field.

The algorithm for computing compact representations in [15] was previously not implemented, and only a rough complexity analysis is available.

1.2 Summary of research contributions

The main result of this thesis is the development of two novel algorithms for solving norm equations using compact representations; one is an exhaustive search algorithm inspired by the existing method by Gaál and Pohst in [21], and the other is an algorithm using principal ideal testing via index calculus. Incorporating compact representations into the algorithms not only allowed for smaller representations of solutions, but also improved the algorithms' efficiency greatly compared to the existing method that is not using compact representations.

The complexity analyses of the novel algorithms and the Gaál-Pohst algorithm for solving norm equations were conducted and compared. In addition to the complexity analyses, we also performed extensive numerical analyses of the algorithms to compare their performance in practice. Our novel algorithms outperformed the Gaál-Pohst method remarkably in terms of time efficiency, and had better asymptotic complexities. The algorithm using principal ideal testing was the most efficient both in terms of asymptotic complexity and in practice,

especially for large inputs.

As a part of our algorithms, we implemented the algorithm for computing compact representations in [15] and analyzed its complexity in detail. In addition, we developed and implemented algorithms for arithmetic operations on compact representations that are necessary for our novel algorithms. All Magma implementation and testing code can be found at https://github.com/s-leem/FF_NormEq_CR.

The following subsections summarize our contributions regarding different topics; analyses of the Gaál-Pohst algorithm in Section 1.2.1, analyses and algorithms related to compact representations in Section 1.2.2, our first novel algorithm inspired by the Gaál-Pohst algorithm in Section 1.2.3, and our second novel algorithm using principal ideal testing in Section 1.2.4.

1.2.1 Analyses of the existing method

This thesis provides an explicit algorithmic form of the Gaál-Pohst method in [21] for solving norm equations, along with a complexity analysis and empirical analysis.

This method is an exhaustive search that collects solutions of a norm equation by testing all elements of F that are represented with respect to a reduced basis, and satisfy degree bounds on the coefficient polynomials. Since the bounds were not explicitly given in [21], we specified the degree bounds of the coefficients $\lambda_i(x)$ and used them to establish a search space. This thesis presents the entire process of the Gaál-Pohst method in an algorithmic form, suitable for a complexity analysis as well as implementation. Moreover, a step-by-step example is also provided for illustrative purposes.

We analyzed the computational complexity of the Gaál-Pohst algorithm by counting the number of bit operations needed to solve a norm equation. We counted the number of elements in the search space and the number of bit operations needed to perform each step of the algorithm, which were used to calculate the total number of bit operations for the entire process. The number of elements in the search space of this algorithm is doubly

exponential in the extension degree n of $F/k(x)$ and the genus g of F, exponential in $\deg c$, and superexponential in q, the size of k. The asymptotic complexity of this algorithm is doubly exponential in n and g, exponential in $\deg c$, and superexponential in q.

Finally, we implemented the algorithm in Magma and conducted numerical experiments. The algorithm was thoroughly tested to verify its correctness, and timed with different parameters. The timing results substantiated the complexity analysis. As expected from the complexity analysis, the algorithm was able to solve norm equations in a reasonable amount of time only when all parameters are very small. For example, it already took more than two days for Gaál-Pohst algorithm to solve a norm equation over $F/\mathbb{F}_q$ when the extension degree of $F/\mathbb{F}_q(x)$ is 2 and $q = 7$, while the newly developed algorithms took less than 1 CPU second for the same equations of $q = 7$, and still took less than a day for a 13-bit prime q.

1.2.2 Analyses and algorithms related to compact representations

Algorithms for conducting arithmetic on compact representations are provided in this thesis. In addition, we provide a detailed complexity analysis of the algorithm for computing compact representations in [15] and an implementation of the algorithm, together with examples.

We developed algorithms to conduct arithmetic of elements $\alpha \in F$ represented in compact representations, including algorithms multiplying two elements, computing an integer power of α, the norm of α, and the value of α at a given place. These algorithms are essential arithmetic for computations using compact representations and are used in our novel algorithms for solving norm equations. A complexity analysis of the algorithms are also provided.

For the algorithm for computing a compact representation in [15], Eisenträger and Hallgren only provided a rough analysis of its complexity. We carefully counted the number of bit operations needed to compute a compact representation. In addition, we analyzed the size of an element of F in compact representation by counting the number of field elements

required to represent it. This result was used to analyze the computational complexity of our new algorithms for solving norm equations.

We implemented the algorithm for computing compact representations in [15] and the algorithms for arithmetic on compact representations in Magma. Thorough tests on these algorithms were performed to confirm that they are correct.

1.2.3 Exhaustive search algorithm using compact representations

The first novel algorithm is inspired by the method of Gaál-Pohst and incorporates compact representations. Similar to the Gaál-Pohst method, this algorithm is an exhaustive search. The search space of this algorithm is notably smaller than Gaál-Pohst's and this algorithm returns solutions in compact representation, which tend to be smaller than the representation Gaál-Pohst used.

The number of elements in the search space of this new algorithm is significantly less than Gaál-Pohst's. Our search space is defined by bounds on the values of a potential solution at places in an appropriate finite set of places, instead of bounds on the degrees of the coefficient polynomials. Elements in the search space are enumerated in compact representations. Similar to the Gaál-Pohst algorithm, our algorithm computes the norm of each element in the search space and collects all elements of norm c up to a non-zero constant. Using compact representations makes computing the norms of elements more efficient as well as representing solutions in smaller sizes.

We conducted a complexity analysis of this algorithm by counting the number of bit operations required to solve a norm equation. The number of elements in the search space of this algorithm is superexponential in n, exponential in g and $\deg c$, and polynomial in q. This algorithm has asymptotic complexity that is superexponential but less than doubly exponential in n, exponential in g and $\deg c$, and subexponential in q. With the smaller search space and computational efficiency of compact representations, the difference between this algorithm and Gaál-Pohst original algorithm in asymptotic complexity was doubly exponential

in n, g, and q.

Finally, we developed an implementation in Magma and performed numerical experiments for the algorithm. Thorough black-box and white-box tests were conducted to verify the correctness. The algorithm was timed with various parameters. The timing results were well aligned with asymptotic complexities. This algorithm was faster on all tested inputs compared to Gaál-Pohst original algorithm, especially on large parameters. For example, when the extension degree of F is 2, it took a similar amount of time to solve norm equations for this algorithm and Gaál-Pohst's original algorithm, however, when $n = 7$, it only took 1.685 CPU seconds for our novel algorithm, while Gaál-Pohst original algorithm took more than a day to solve a norm equation. Further details on the testing parameters are in Chapter 5.

Note that to use the algorithm for computing compact representations in [15], it is required for F to have at least one infinite place of degree 1. Since there are infinitely many global function fields having a degree 1 infinite place, this technique can be applied broadly. In addition, for F with no infinite place of degree 1, defining a search space using valuations instead of degrees of coefficients still yields a smaller search space.

1.2.4 Principal ideal generator approach

The second novel algorithm uses index calculus to conduct principal ideal tests and also incorporates compact representations. This algorithm searches for principal ideals of norm c up to non-zero constants in k and computes generators of them. Note that one can find elements of certain norms by computing generators of principal ideals of the same norm.

The new algorithm first enumerates all ideals I that divide the principal ideal generated by c, computes the norm of each I, and conducts principal ideal tests only on I of the norm c up to a non-zero constant. When performing principal ideal tests, this algorithm uses the fact that a solution α of (1.1) is an S_c-unit with a certain finite set S_c of places. Here, α can be written as $\alpha = \prod_1^{|S_c|-1} {\epsilon_i}^{x_i}$, where $\{\epsilon_i | 1 \leq i \leq |S_c| - 1\}$ is a system of fundamental

S_c-units.

We designed a principal ideal test for our algorithm, which is different from the existing principal ideal tests using a randomized relation search in [25]. Since we know how I factors, our algorithm does not search for random relations but solves a certain matrix equation that computes x_i for $\alpha = \prod_1^{|S_c|-1} {\epsilon_i}^{x_i}$, which is a deterministic process. If the matrix equation is consistent, I is principal. Using a solution of the matrix equation we compute the values of a generator α of I at the infinite places of F. Finally, a compact representation of the generator α is computed with inputs I and the values of α at the infinite places. Compact representations make it possible to represent the solutions in smaller sizes.

A rigorous complexity analysis of this algorithm was conducted by counting the number of bit operations needed to solve a norm equation. With the help of using compact representations and the idea of searching ideals instead of elements, this algorithm has a much smaller asymptotic complexity compared to the original Gaál-Pohst algorithm and our exhaustive search algorithm using compact representations. The number of ideals to search in this algorithm is exponential in n and $\deg c$, and does not depend on g and q. The asymptotic complexity of this algorithm is exponential in n and $\deg c$ and subexponential in g and q. Compared to Gaál-Pohst's original algorithm, it is doubly exponentially smaller in n, g, and q. It is smaller by exponential factors in n and g, and a polynomial factor in q, even compared to the exhaustive search algorithm using compact representations in Section 1.2.3.

We implemented and tested this algorithm in Magma and thoroughly tested this algorithm also using black-box white-box methodology to verify its correctness. Then the algorithm was timed with various parameters. This algorithm was the fastest for all tested inputs compared to the other two algorithms. In addition, it solved norm equations exceptionally faster than the other two algorithms when F has a high genus. For example, for norm equations over F with genus 9, this algorithm only took 0.63 CPU seconds to solve a norm equation on average, while it took more than a day to solve them using the other

algorithms. This algorithm started to take more than a day from when $\deg c$ is a 16-bit prime, while the original Gaál-Pohst algorithm took more than a day from $\deg c = 13$ with smaller fixed parameters. Detailed testing information can be found in Chapter 5.

As mentioned in Section 1.2.3, in order to use the algorithm for computing compact representations in [15], F must have at least one infinite place of degree 1. However, it is still possible to search for principal ideals of norm c in the same way and compute a generator in other representations even if F has no such infinite place.

1.3 Outline of the thesis

In this section, we provide an outline of the thesis.

Chapter 2 provides preliminary material on algebraic function fields and compact representations. First, Section 2.1 presents definitions and properties of algebraic function fields including discrete valuations, ideals, and divisors. Later in the section, the definition of norm equations over function fields is provided. Section 2.2 summarizes the definitions and properties of lattices and normed spaces, along with examples in the context of algebraic function fields. In Section 2.3, we analyze the computational costs of fundamental arithmetic in global function fields, including arithmetic with elements and ideals of global function fields. Section 2.4 first summarizes the algorithm for computing compact representations of [15], then provides algorithms conducting various arithmetic with compact representations and detailed computational complexity analyses of the algorithms.

Chapter 3 begins with a summary of the approach for solving norm equations in O_F of Gaál and Pohst in [21]. Section 3.1.1 begins with a summary of the approach, followed by an explicit description of the search space as well as an example. Section 3.1.2 analyzes the number of elements in the search space and the number of bit operations required to solve a norm equation. A discussion on the asymptotic complexity of the algorithm is also provided at the end of the section. In Section 3.2, we introduce a new algorithm that is inspired

by Gaál and Pohst's approach. Section 3.2.1 introduces the algorithm along with a way to enumerate elements using compact representations. Section 3.2.2 analyzes the number of elements in the search space of the new algorithm and the number of bit operations required to solve a norm equation. An asymptotic complexity of the new algorithm and a comparison to the asymptotic complexities and the sizes of the solutions to the existing algorithm are also provided in this section.

In Chapter 4, we introduce an index calculus approach for solving norm equations in O_F along with examples, which is related to the principal ideal problem. Section 4.1 contains a description of a new algorithm for solving norm equations and its complexity. Section 4.1.1 first provides a deterministic principal ideal test by solving a matrix equation. Then we show how to find all principal ideals of the norm c and compute compact representations of their generators. At the end of Section 4.1.1, a brief description and an example of a variation of the algorithm using S-units are provided. Finally, Section 4.1.2 provides the number of ideals to search and the number of bit operations required to solve a norm equation. The computational cost of the variation is also provided at the end of the section.

Chapter 5 provides numerical timing experiments for the existing and new algorithms for solving norm equations. All implementation and tests were conducted in Magma. We provide timing test results of the algorithms for varying n, g, q, $\deg c$, and the number of distinct factors of c. We also compare the test results to our theoretical complexity analysis results. The results in this chapter provide evidence supporting the comparison of complexities of the algorithms described in the previous chapters.

A brief summary of this thesis and open problems are given in Chapter 6. All novel implementations were developed by the author of this thesis.

Chapter 2

Algebraic function fields and compact representations

This chapter covers the necessary definitions and properties of an algebraic function field F and related topics for the other chapters of this thesis. In Section 2.1, we define preliminary objects such as algebraic function fields F/k, height functions in $k(x)$ and F, and the norm of $\alpha \in F$. Then in Section 2.1.1, we describe how valuations and places are defined in F, along with the finite maximal order O_F and the infinite maximal order $O_{F,\infty}$ of F. Next, in Section 2.1.2, we focus on definitions and properties related to ideals of O_F and divisors of F. In Section 2.2, we define lattices and normed spaces and view O_F and F as a lattice and a normed space, respectively. We also define S-unit lattices of F and provide an algorithm for computing them. We analyze the computational cost of performing fundamental arithmetic in function fields; including the cost of multiplication and exponentiation of polynomials, rationals, and elements of F in Section 2.3. Finally, in Section 2.4, we present two ways to represent $\alpha \in F$; the standard representation and a compact representation. We first summarize algorithms to compute a compact representation of elements α as in [15] in Section 2.4.1. Then we develop algorithms to conduct arithmetic operations with compact representations in Section 2.4.2. Finally, we perform complexity analyses on the algorithms

in Section 2.4.3.

2.1 Introduction to algebraic function fields

In this section, we first define an algebraic function field F/k. Then we define the height of elements in $k(x)$ and in F. Lastly, we define the norm of an element $\alpha \in F$ and present its properties.

Let k be a field and x be transcendental over k. We denote the polynomial ring over k in x by $k[x]$, and the rational function field over k in x by $k(x)$.

Definition 2.1. [42, Definition 1.1.1] An *algebraic function field* F/k of one variable over k is an extension field $F \supseteq k$ such that F is a finite algebraic extension of $k(x)$ for some element $x \in F$ which is transcendental over k.

The degree of F is the extension degree of F over $k(x)$ and is denoted by n in this thesis. The rational function field $k(x)$ can be considered as an algebraic function field over $k(x)$ of the extension degree 1, and we call $k(x)$ the rational function field over k.

The field k is called the constant field of F/k. The algebraic closure of k in F, which is denoted by k_0, is the full constant field of F/k. The extension F/k is called geometric if $k = k_0$. We say k is algebraically closed in F (or k is the full constant field of F) if $k_0 = k$. We say F/k is a global function field when k is a finite field. When F is a global function field, $F/k(x)$ is a separable extension [42, p.144]. A polynomial $f \in k[x][t]$ is called a defining polynomial of $F = k(x)(y)$ if f is a separable irreducible polynomial and y is a root of f. When $F/k(x)$ is a finite extension, F is a finitely generated $k(x)$-vector space of dimension n. A basis of F is referred to as a $k(x)$-basis of F over $k(x)$. We call $\{y^{i-1} \mid 1 \leq i \leq n\}$ the standard basis of F.

We define the height of a rational function λ in $k(x)$, and of an element α of F with respect to a basis $\mathcal{B} = \{\omega_1, \ldots, \omega_n\}$ as follows. We will use the height mainly in the sections for the complexity analysis.

Definition 2.2. [5, Definition 2.9] For $\lambda = \frac{\lambda_1}{\lambda_2} \in k(x)$, with coprime polynomials $\lambda_1, \lambda_2 \in k[x]$, $\lambda_2 \neq 0$, we define the *height* of λ to be

$$\mathbf{h}(\lambda) := \max\{\deg(\lambda_1), \deg(\lambda_2)\}.$$

Definition 2.3. [6, Definition 3.2] Let $\mathcal{B} = \{\omega_i | 1 \leq i \leq n\}$ be a basis of F, and $\alpha = \sum_{i=1}^{n} \lambda_i \omega_i$ be an element of F, with $\lambda_i \in k(x)$ for $1 \leq i \leq n$. The *height* of α with respect to $\mathcal{B}$ is defined by

$$\mathbf{h}_{\mathcal{B}}(\alpha) := \max_{1 \leq i \leq n} \mathbf{h}(\lambda_i).$$

For simplicity, the height of α with respect to the standard basis $\mathcal{B}_y = \{y^{i-1} \mid 1 \leq i \leq n\}$ of F is denoted by $\mathbf{h}(\alpha)$ instead of $\mathbf{h}_{\mathcal{B}_y}(\alpha)$.

Lastly, we define the norm of an element $\alpha \in F$ with respect to the algebraic extension $F/k(x)$ as follows. The definition and properties of the norm are taken from [42, A.14]. Let $\mathcal{B} = \{\omega_1, \ldots, \omega_n\}$ be a basis of F. For an element $\alpha \in F$, we consider the multiplication-by-α map, which can be written with a matrix $M_\alpha \in F^{n \times n}$,

$$\alpha \begin{bmatrix} \omega_1 & \ldots & \omega_n \end{bmatrix} = \begin{bmatrix} \omega_1 & \ldots & \omega_n \end{bmatrix} M_\alpha. \tag{2.1}$$

Then the norm of α is the determinant of M_α, $\mathrm{Norm}_{F/k(x)}(\alpha) := \det M_\alpha$. The norm map has the following properties:

(1) The norm map is multiplicative; $\mathrm{Norm}_{F/k(x)}(\alpha_1\alpha_2) = \mathrm{Norm}_{F/k(x)}(\alpha_1) \cdot \mathrm{Norm}_{F/k(x)}(\alpha_2)$ for $\alpha_1, \alpha_2 \in F$.

(2) $\mathrm{Norm}_{F/k(x)}(\alpha) = 0$ if and only if $\alpha = 0$

(3) For $\lambda \in k(x)$, $\mathrm{Norm}_{F/k(x)}(\lambda) = \lambda^n$.

(4) $\mathrm{Norm}_{F/k(x)}(\alpha) = \prod_{j=1}^{n} \sigma_j(\alpha) \quad \in k(x)$, where σ_j are embeddings of F into a fixed algebraic closure of F.

(5) $\mathrm{Norm}_{F/k(x)}(\alpha)$ does not depend on the choice of a basis of $F/k(x)$.

2.1.1 Valuations and places

In function fields, places are the equivalent of primes in number fields. Hence, it is important to understand places and valuations. In this section, we first present definitions of a discrete normalized valuation of F, a discrete valuation ring, and a place P in F, along with their properties and examples. Then we describe places in extension fields and define finite and infinite places together with the finite maximal order O_F and the infinite maximal order $O_{F,\infty}$ of F.

In this thesis, we deal with discrete normalized valuations on algebraic function fields F/k, including the rational function field $k(x)$. We first define a discrete normalized valuation on a function field F/k and a valuation ring in F/k.

Definition 2.4. [42, Definition1.1.9] A *discrete normalized valuation* on a function field F/k is a map $v : F \longrightarrow \mathbb{Z} \cup \{\infty\}$ such that for all $\alpha, \beta \in F$:

(1) $v(\alpha) = \infty$ if and only if $\alpha = 0$

(2) $v(\alpha\beta) = v(\alpha) + v(\beta)$

(3) $v(\alpha + \beta) \geq \min\{v(\alpha), v(\beta)\}$

(4) There exists an element $\psi \in F$ with $v_P(\psi) = 1$. Such an element ψ is a uniformizer (or a prime element) for v.

For brevity, we simply call discrete normalized valuations on $k(x)$ and F valuations of $k(x)$ and F from now on.

For example, in $k(x)$, we have the infinite valuation and the p-adic valuations defined as

follows. Let $\lambda = \frac{\lambda_1}{\lambda_2} \in k(x)$ with $\lambda_2 \neq 0$. The infinite valuation v_∞ on $k(x)$ is defined as

$$v_\infty : k(x) \longrightarrow \mathbb{Z} \cup \{\infty\}, \quad v_\infty(\lambda) = \begin{cases} \deg(\lambda_2) - \deg(\lambda_1) & \text{if } \lambda \neq 0, \\ \infty & \text{if } \lambda = 0. \end{cases} \tag{2.2}$$

The p-adic valuations v_p on $k(x)$ are an analogue of the p-adic valuations on the set of rational numbers $\mathbb{Q}$ and are defined as follows. Let $p(x) \in k[x]$ be a monic irreducible polynomial. Then $\lambda \in k(x)$ can be written in the form $\lambda = \frac{\lambda_1'(x)}{\lambda_2'(x)} p(x)^m$ where $m \in \mathbb{Z}$, $\lambda_2'(x) \neq 0$, $p(x) \nmid \lambda_1'(x)\lambda_2'(x)$. The map

$$v_p : k(x) \longrightarrow \mathbb{Z} \cup \{\infty\}, \quad (0 \neq)\lambda \mapsto m, \text{ and } 0 \mapsto \infty$$

is a p-adic valuation on $k(x)$. In fact, there is no valuation in $k(x)$ that is not the infinite valuation nor a p-adic valuation [42, Theorem 1.2.2].

A discrete valuation ring and a place of F, which are closely related to valuations on F, are defined as follows.

Definition 2.5. [42, Definition 1.1.4] A *discrete valuation ring* of the function field F/k is a ring $O \subseteq F$ with the following properties:

(1) $k \subsetneqq O \subsetneqq F$, and

(2) for every $\alpha \in F^\times$ we have $\alpha \in O$ or $\alpha^{-1} \in O$.

For brevity, we will henceforth refer to a discrete valuation ring of F simply as a valuation ring F.

We denote the unit group of O by $O^\times = \{\alpha \in F^\times \mid \alpha \in O \text{ and } \alpha^{-1} \in O\}$. Any valuation ring of F satisfies the following proposition.

Proposition 2.6. *[42, Proposition 1.1.5, Theorem 1.1.6] Let O be a valuation ring of a function field F, and let $O^\times$ be the unit group of O. Then the following hold:*

(1) O is a local ring; i.e., O has a unique maximal ideal $P = O \setminus O^\times$, where $O^\times = \{\alpha \in O \mid$ there is an element $\beta \in O$ with $\alpha\beta = 1\}$ is the group of units of O.

(2) Let $0 \neq \alpha \in P$. Then $\alpha \in P \iff \alpha^{-1} \in O$.

(3) For the full constant field k_0 of F/k, we have $k_0 \subseteq O$ and $k_0 \cap P = \{0\}$.

(4) P is a principal ideal.

(5) If $P = \psi O$ then each $0 \neq \alpha \in F$ has a unique representation of the form $\alpha = \psi^m u$ for some $m \in \mathbb{Z}$ and $u \in O^\times$.

(6) O is a principal ideal domain. More precisely, if $P = \psi O$ and $\{0\} \neq I \subseteq O$ is an ideal then $I = \psi^m O$ for some positive integer $m \in \mathbb{Z}$.

Definition 2.7. [42, Definition 1.1.8] A *place* P of the function field F/k is the maximal ideal of some valuation ring O of F/k. Every element $\psi \in P$ such that $P = \psi O$ is called a *prime element* for P. We denote the set of all places of F/k by $\mathbb{P}(F)$.

There is a one-to-one correspondence between the set of valuations of F with $v(a) = 0$ for all $a \in k^\times$ and the set $\mathbb{P}(F)$ of places of F [42, Theorem 1.1.13]. We denote the valuation corresponding to a place P by v_P and call v_P the valuation associated to P or the valuation at P. Let $\alpha \in F$. We call $v_P(\alpha)$ the value of α at P. For a valuation v on a function field F/k, we can define the following subsets of F:

$$\begin{aligned} O_v &= \{\alpha \in F \mid v(\alpha) \geq 0\}, \\ O_v^\times &= \{\alpha \in F \mid v(\alpha) = 0\}, \text{ and} \\ P_v &= \{\alpha \in F \mid v(\alpha) > 0\} = O_v \setminus O_v^\times. \end{aligned}$$

In fact, O_v is a valuation ring of F, $O_v^\times$ is the unit group of O_v, and P_v is the unique maximal ideal of O_v, so P_v is a place of F.

For example, in the rational function field $k(x)$, every p-adic valuation on $k(x)$ with a monic irreducible polynomial $p(x) \in k[x]$ has an associated place. To be specific,

$$O_{p(x)} = \left\{ \frac{\lambda_1(x)}{\lambda_2(x)} \in k(x) \;\middle|\; \lambda_1(x) \in k[x],\ \lambda_2(x) \in k[x] \setminus \{0\},\ p(x) \nmid \lambda_2(x) \right\}$$

is the valuation ring with the maximal ideal

$$p_{p(x)} = \left\{ \frac{\lambda_1(x)}{\lambda_2(x)} \in k(x) \;\middle|\; \lambda_1(x) \in k[x],\ \lambda_2(x) \in k[x] \setminus \{0\},\ p(x) \mid \lambda_1(x), p(x) \nmid \lambda_2(x) \right\},$$

which is a place of $k(x)$ [42, p.2]. In addition, v_∞ has the associated place denoted by p_∞ with the valuation ring

$$O_{p_\infty} = \left\{ \frac{\lambda_1(x)}{\lambda_2(x)} \in k(x) \;\middle|\; \lambda_1(x) \in k[x],\ \lambda_2(x) \in k[x] \setminus \{0\}, \deg(\lambda_1) \leq \deg(\lambda_2) \right\},$$

and the maximal ideal P_{p_∞} of O_{p_∞}

$$P_{p_\infty} = \left\{ \frac{\lambda_1(x)}{\lambda_2(x)} \in k(x) \;\middle|\; \lambda_1(x) \in k[x],\ \lambda_2(x) \in k[x] \setminus \{0\}, \deg(\lambda_1) < \deg(\lambda_2) \right\}.$$

We often denote O_{p_∞} and P_{p_∞} by O_∞ and P_∞ for brevity.

Let $P \in \mathbb{P}(F)$. We define the residue class field F_P of P as $F_P := O_P/P$, and the degree of P as in [42, Definition 1.1.14],

$$\deg P = [F_P : k].$$

We call a degree 1 place a rational place of F/k. For any $P \in \mathbb{P}(F)$, the degree of P is finite [42, Proposition 1.1.15].

Let v_P be the valuation at a place P. If $v_P(\alpha) > 0$, we say that P is a zero of α. If $v_P(\alpha) < 0$, we say that P is a pole of α. Although F has infinitely many places, any $\alpha \in F$ has only finitely many zeros and poles [42, Corollary 1.3.2, Corollary 1.3.4]. In addition, the zeros of α satisfy the following inequality. Let $P_1, \ldots P_m$ be all the zeros of $\alpha \in F$. Then by

[42, Proposition 1.3.3],

$$\sum_{i=1}^{m} v_{P_i}(\alpha) \deg P_i \leq [F : k(\alpha)] \leq [F : k(x)] = n.$$

Valuations and places can also be considered in an algebraic extension field F'/k' of F/k, which is defined as follows.

Definition 2.8. [42, Definition 3.1.1] An algebraic function field F'/k' is called an algebraic extension of F/k if $F' \supseteq F$ is an algebraic field extension and $k' \supseteq k$. In addition, the algebraic extension F'/k' of F/k is called a finite extension if $[F' : F] < \infty$.

Let F'/k' be a finite algebraic extension of F/k. We say a place $P' \in \mathbb{P}(F')$ lies above $P \in \mathbb{P}(F)$ or is an extension of P if $P \subseteq P'$ [42, Definition 3.1.3]. We denote this by $P'|P$. Places P' of F'/k' have the following properties.

Proposition 2.9. *[42, Proposition 3.1.4] Let F'/k' be a finite algebraic extension of F/k. Suppose that P is a place of F/k and P' is a place of F'/k', and let $O_P \subseteq F$ and $O_{P'} \subseteq F'$ denote the corresponding valuation rings of the valuations v_P of F/k and $v_{P'}$ of F'/k', respectively. Then the following assertions are equivalent:*

(a) $P'|P$

(b) $O_P \subseteq O_{P'}$

(c) There exists an integer $e \geq 1$ such that $v_{P'}(\alpha) = e v_P(\alpha)$ for all $\alpha \in F$.

Moreover, if $P'|P$ then $P = P' \cap F$ and $O_P = O_{P'} \cap F$.

Proposition 2.10. *[42, Proposition 3.1.7] Let F'/k' be a finite algebraic extension of F/k.*

(1) For each place $P' \in \mathbb{P}(F')$, there is exactly one place $P \in \mathbb{P}(F)$ such that $P'|P$.

(2) Conversely, every place $P \in \mathbb{P}(F)$ has at least one, but only finitely many extensions $P' \in \mathbb{P}(F')$

The positive integer e in part (c) of Proposition 2.9 is called the ramification index of $P'|P$, and is often denoted by $e_{P'|P}$ [42, Definition 3.1.5 (a)]. When $e_{P'|P} > 1$, $P'|P$ is said to be ramified, and when $e_{P'|P} = 1$, $P'|P$ is said to be unramified. In addition, we define the residue (or relative) degree $f_{P'|P}$ of $P'|P$ by $f_{P'|P} := [F'_{P'} : F_P]$ [42, Definition 3.1.5 (b)]. In this thesis, we consider this definition in the context that F/k is an algebraic extension of the rational function field $k(x)$. In this case, for $P|p$ where $P \in \mathbb{P}(F)$ and $p \in \mathbb{P}(k(x))$, since we have

$$\deg P = [F_P : k] = [F_P : F_p][F_p : k] = f_{P|p} \cdot \deg p,$$

the relative residue degree of $P|p$ is $f_{P|p} = [F_P : F_p] = \deg P / \deg p$.

Let P be a place of F/k and $P'_1, \ldots, P'_m$ be all the places of F'/k' lying above P. Let e_i denote the ramification index $e_{P'_i|P}$ and f_i the relative residue degree of $P'_i|p$. We have the Fundamental Equality as follows [42, Theorem 3.1.11].

$$\sum_{i=1}^{m} e_i f_i = [F' : F]. \tag{2.3}$$

Now we consider the finite and infinite maximal orders of F/k. The finite maximal order O_F of F is the integral closure of $k[x]$ in F [3, p.6]. The finite maximal order O_F is a finitely generated $k[x]$-module [38, p.13 Proposition 8]. In this thesis, a basis of O_F means a $k[x]$-basis of O_F. The infinite maximal order $O_{F,\infty}$ of F is defined to be the integral closure of O_{p_∞} in F [3, p.6].

Let $P \in \mathbb{P}(F)$. We call P a finite place of F if it lies above $p_{p(x)}$ for some monic irreducible polynomial $p(x) \in k[x]$. In this case, $\mathfrak{p} := P \cap O_F$ is a prime ideal of O_F, and we call $\mathfrak{p}$ the prime ideal corresponding to the place P. On the other hand, we call P an infinite place of F if P lies above p_∞. In this case, $\mathfrak{p} := P \cap O_{F,\infty}$ is a prime ideal of $O_{F,\infty}$, and we call $\mathfrak{p}$ the prime ideal corresponding to the place P. The sets of finite places and infinite places of F are denoted by $\mathbb{P}_0(F)$ and $\mathbb{P}_\infty(F)$, respectively. Both O_F and $O_{F,\infty}$ are Dedekind domains.

The unit group $O_F^\times$ of O_F is defined by

$$O_F^\times := \{\alpha \in O_F | \alpha^{-1} \in O_F\}.$$

Each element of $O_F^\times$ is a unit of O_F, and a set of units that generates the torsion-free part of $O_F^\times$ is called a set of fundamental units. The unit group can be generalized to S-unit group which will be discussed later in Section 2.2.2.

2.1.2 Ideals and divisors

In this section, we discuss the definitions and properties of ideals of O_F and divisors of F. Later in this section, we define reduced ideals in O_F and present a bound on the degree of the divisor of a reduced ideal at the end of this section, which will be used in Section 2.3. We keep the same notation from the previous section.

We define a fractional ideal of O_F as a finitely generated nonzero O_F-submodule of F of rank n [43, p.10]. Throughout this thesis, an ideal of O_F is called an integral ideal of O_F, and a fractional O_F-ideal is called an ideal of O_F for simplicity. The set of nonzero fractional ideals of O_F is denoted by $\mathrm{Id}(O_F)$. The group $\mathrm{Id}(O_F)$ is an abelian group under multiplication [43, Theorem 5]. When an ideal $I \in \mathrm{Id}(O_F)$ is generated by one element of F, we say I is a principal ideal of O_F. The group of nonzero principal ideals of O_F is a subgroup of $\mathrm{Id}(O_F)$ and denoted by $\mathrm{PrincId}(O_F)$. The ideal class group $Cl(O_F)$ of O_F is defined by $Cl(O_F) := \mathrm{Id}(O_F)/\mathrm{PrincId}(O_F)$ [43, p.15]. The ideal class group $Cl(O_F)$ is a finite abelian group, and we call the order of $Cl(O_F)$ the ideal class number of O_F, and denote it by h_{O_F}.

Let I be an ideal of O_F. We define the prime ideal factorization of I, the value of I at a prime ideal $\mathfrak{p}$, the discriminant of I, norm of I in the order.

Lemma 2.11. *[43, Corollary to Theorem 6] Let O_F be the finite maximal order of a function*

field F/k. Any nonzero ideal I can be written in the form

$$I = \prod_{i=1}^{m} \mathfrak{p}_i^{m_i},$$

where the $\mathfrak{p}_i$ are distinct prime ideals, each corresponding to a place P and $m_i \in \mathbb{Z} \setminus \{0\}$ for $1 \leq i \leq m$.

With the ideal factorization in the lemma above, we define the value of an ideal I at a place P by $v_{\mathfrak{p}}(I) = m_i$, where $\mathfrak{p}$ is the prime ideal corresponding to P.

Now we define the discriminant of I. Note that I is a $k[x]$-module of rank n [44, Theorem 4.1.2]. Let $\mathcal{B} = \{\omega_1, \ldots, \omega_n\}$ be a $k[x]$-basis of I. Then the discriminant of I is denoted by $\Delta(I)$ and defined by

$$\Delta(I) = \left(\det \begin{bmatrix} \omega_1^{(1)} & \omega_2^{(1)} & \ldots & \omega_n^{(1)} \\ \omega_1^{(2)} & \omega_2^{(2)} & \ldots & \omega_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \omega_1^{(n)} & \omega_2^{(n)} & \ldots & \omega_n^{(n)} \end{bmatrix} \right)^2,$$

where $\omega_i^{(j)} = \sigma_j(\omega_i)$ is the j-th conjugate of ω_i in the extension $F/k(x)$ [44, p.75]. The discriminant of I does not depend on the choice of $\mathcal{B}$ [44, Theorem 4.1.2].

The norm of I is defined as follows.

Definition 2.12. [44, Definition 4.1.4] Let I be an ideal of O_F. Then *the norm of I* is defined and denoted by

$$\mathrm{Norm}(I) = \sqrt{\frac{\Delta(I)}{\Delta(O_F)}} \in k(x).$$

If we let $\{\omega_1, \ldots, \omega_n\}$, $\{\alpha_1, \ldots, \alpha_n\}$ be $k[x]$-bases of O_F and I, respectively, then equivalently, $\mathrm{Norm}(I)$ is defined to be the determinant of $T \in GL(n, k(x))$ such that

$$\begin{bmatrix} \alpha_1 & \ldots & \alpha_n \end{bmatrix} = \begin{bmatrix} \omega_1 & \ldots & \omega_n \end{bmatrix} T.$$

Since $\Delta(I)$ and $\Delta(O_F)$ are defined up to nonzero constant square factors in k, $\mathrm{Norm}(I)$

is also defined up to a nonzero constant factor in k. In addition, the norm of I satisfies the following properties.

Lemma 2.13. *Let F/k be a function field and O_F be the finite maximal order of F. Then the following holds.*

(1) Let $\alpha \in F$. Then $Norm(\alpha O_F)/Norm_{F/k(x)}(\alpha) \in k^\times$

(2) Let I_1, I_2 be ideals of O_F. Then $Norm(I_1 I_2) = Norm(I_1)Norm(I_2)$.

Proof. (1) We know $\mathrm{Norm}_{F/k(x)}(\alpha) = \det M_\alpha$, where M_α is the matrix defined in (2.1). Since $\{\alpha\omega_1, \ldots, \alpha\omega_n\}$ is a basis of αO_F, $\mathrm{Norm}(\alpha O_F)$ is $\mathrm{Norm}_{F/k(x)}(\alpha)$ up to a constant factor. For (2), see [1, Theorem 9.3.2]. □

We represent an ideal of O_F in the Hermite normal form (HNF) representation.

To define the HNF representation of an ideal I, we need to define the denominator $d(I)$ of I, an HNF matrix with polynomial entries, and the HNF basis of an integral ideal I_0.

The denominator $d(I)$ of I is the unique monic polynomial in $k[x]$ of minimal degree such that $d(I)I$ is an integral ideal of O_F [44, p.147]. The denominator of I satisfies the following properties.

Lemma 2.14. *[44, Proposition 5.1.2] Let I_1 and I_2 be ideals of O_F. Then*

$$d(I_1 I_2) \mid d(I_1)d(I_2).$$

Consequently, $\deg d(I_1 I_2) \leq \deg d(I_1) + \deg d(I_2)$.

An HNF matrix with polynomial entries is analogous to an HNF matrix with integer entries. A polynomial matrix in HNF is defined as follows.

Definition 2.15. [44, Definition 5.1.6] An $m \times n$ matrix M whose (i, j)-th entry is $a_{i,j} \in k[x]$ for $1 \leq i \leq m$ and $1 \leq j \leq n$ is said to be a *Hermite normal form (HNF) matrix* (or M is in HNF) if there exists a non-negative integer $r \leq n$ and a strictly increasing map $f : \{r+1, \ldots, n\} \to \{1, \ldots, m\}$ such that

1. The first r columns of M are equal to 0.

2. For $r+1 \leq j \leq n$, $a_{f(j),j} \neq 0$ and is monic, and the entries of j^{th} column satisfy the following properties: $a_{i,j} = 0$ if $i > f(j)$ and $\deg a_{f(j),j} > \deg a_{f(j),l}$ for all $j+1 \leq l \leq n$.

With the definition of a HNF $k[x]$-basis of an integral ideal I_0 as follows. Let I_0 be an integral ideal of O_F, $\mathcal{B}_{I_0} = \{\alpha_1, \ldots, \alpha_n\}$ be its $k[x]$-basis, and $\{\omega_1, \ldots, \omega_n\}$ be a fixed $k[x]$-basis of O_F. Write $\alpha_i := \sum_{j=1}^{n} a_{i,j}\omega_j$. The coefficient matrix M_{I_0} of $\mathcal{B}_{I_0}$ is defined by the coefficients of α_i; $(M_{I_0})_{i,j} := a_{i,j}$ for $1 \leq i, j \leq n$. We call $\mathcal{B}_{I_0}$ the HNF basis of I_0 if M_{I_0} is in HNF. Any integral ideal I_0 of O_F is known to have an HNF $k[x]$-basis, and the basis depends on the fixed integral basis $\{\omega_1, \ldots, \omega_n\}$ [44, p.158].

Now we can define the HNF representation of an ideal I.

Definition 2.16. [44, Definition 5.1.14] Let I be a fractional ideal with denominator $d(I)$, let $\mathcal{B}_{I_0} = \{\alpha_1, \ldots, \alpha_n\}$ be the $k[x]$-basis of $I_0 := d(I)I$, and let M_{I_0} be a coefficient matrix of $\mathcal{B}_{I_0}$ in HNF. Then the pair $(M_{I_0}, d(I))$ is called the *HNF representation of* I. We write $I = (M_{I_0}, d(I))$ for the HNF representation of I.

Note that when an ideal I of O_F is given in HNF representation $(M_{I_0}, d(I))$, from [44, Proposition 5.1.17], we have

$$\max_{1 \leq i,j \leq n} \{\deg(M_{I_0})_{i,j}\} \leq \deg \mathrm{Norm}(d(I)I) \qquad \text{and} \qquad \mathrm{Norm}(I) = \frac{\det M_{I_0}}{d(I)^n}. \tag{2.4}$$

In later chapters, we assume that ideals are given in HNF representation.

Next, we define divisors, the divisor group of F, the support of a divisor, and effective divisors as in [42, Definition 1.4.1]. The divisor group of F is the additive free abelian group generated by the places of F/k, and is denoted by $Div(F)$. The elements of $Div(F)$ are called divisors of F. A divisor D of F/k is defined as a formal sum

$$D = \sum_{P \in \mathbb{P}(F)} v_P(D)P \text{ with } v_P(D) \in \mathbb{Z}, \text{ only finitely many } v_P(D) \neq 0.$$

Each coefficient $v_P(D)$ is called the value of D at P [42, p.16].

Since $Div(F)$ is a group, we have addition and an additive identity defined in $Div(F)$ [42, Definition 1.4.1]. Given two divisors $D = \sum_{P\in\mathbb{P}(F)} v_P(D)P$ and $D' = \sum_{P\in\mathbb{P}(F)} v_P(D')P$, we can add them by adding their coefficients of each $P \in \mathbb{P}(F)$,

$$D + D' = \sum_{P\in\mathbb{P}(F)} (v_P(D) + v_P(D'))P.$$

The additive identity of $Div(F)$ is called the zero divisor, $0 := \sum_{P\in\mathbb{P}(F)} 0 \cdot P$. In addition, when given two divisors D and D', we say $D \leq D'$ if $v_P(D) \leq v_P(D')$ for all $P \in \mathbb{P}(F)$. When compared to the zero divisor, if $D \geq 0$, meaning that $v_P(D) \geq 0$ for all $P \in \mathbb{P}(F)$, we say D is an effective divisor [42, p.16].

The support of D is the set of P such that $v_P(D)$ is a nonzero integer. The degree of D is defined as the homomorphism $\deg : Div(F) \longrightarrow \mathbb{Z}$ given by

$$\deg D = \sum_{P\in\mathbb{P}(F)} v_P(D) \deg P.$$

For any $D \in Div(F)$, the support of D is a finite set and $\deg D$ is also finite [42, p.15]. The set of divisors of degree 0 is a subgroup of $Div(F)$, we denote it by $Div^0(F)$ [42, Definition 5.1.2].

The height of D is defined with the degree of D as follows [3, Definition 4.1.2]:

$$\mathbf{h}(D) := \deg(D'),$$

where $D' = \sum_{P\in\mathbb{P}(F)} |v_P(D)|P$

Divisors and ideals are closely related. We have homomorphisms defined between $Div(F)$

and $\mathrm{Id}(O_F)$ as follows. $\phi : Div(F) \longrightarrow \mathrm{Id}(O_F)$ is a homomorphism defined by

$$\phi\left(\sum_{P\in\mathbb{P}(F)} v_P(D)P\right) = \prod_{P\in\mathbb{P}_0(F)} \mathfrak{p}^{-v_P(D)}, \tag{2.5}$$

where $\mathfrak{p}$ is the prime ideal of O_F corresponding to the place P [15, p.341]. The right inverse of ϕ is also a homomorphism, $\mathrm{div} : \mathrm{Id}(O_F) \longrightarrow Div(F)$ defined by

$$\mathrm{div}\left(\prod_{P\in\mathbb{P}_0(F)} \mathfrak{p}^{n_P}\right) = -\sum_{P\in\mathbb{P}_0(F)} n_P P,$$

where $n_P \in \mathbb{Z}$. Similarly, a map from $F^\times$ to $Div(F)$ can be defined. Let $\alpha \in F^\times$. The map $\mathrm{div} : F^\times \longrightarrow Div(F)$ is defined by

$$\mathrm{div}(\alpha) = \sum_{P\in\mathbb{P}(F)} v_P(\alpha)P.$$

If a divisor D has an element $\alpha \in F^\times$ such that $\mathrm{div}(\alpha) = D$, we call D a principal divisor. The set of principal divisors of F is a subgroup of $Div(F)$ and is denoted by $\mathrm{PrincDiv}(F)$ [42, Definition 1.4.3].

Let α be a nonzero element of F and $D = \mathrm{div}(\alpha)$. The zero divisor D_0 of α and the pole divisor D_∞ of α are defined as follows. Let Z be the set of zeros of α, and N be the set of poles of α. Then

$$D_0 := \sum_{P\in Z} v_P(\alpha)P$$
$$D_\infty := \sum_{P\in N} -v_P(\alpha)P$$

Clearly, $D_0 \geq 0$, $D_\infty \geq 0$, and $D = D_0 - D_\infty$ [42, Definition 1.4.2]. In addition, for any nonzero $\alpha \in F$, D_0 and D_∞ have the same degree, and we have $\deg D_0 = \deg D_\infty = [F : k(\alpha)]$, thus $\deg D = 0$ [42, Theorem 1.4.11]. This shows $\mathrm{PrincDiv}(F)$ is a subgroup of

$Div^0(F)$. We define the divisor class group $Cl(F) := Div(F)/\mathrm{PrincDiv}(F)$ [42, Definition 1.4.3]. For any D in $Div(F)$, we call $[D] = D + \mathrm{PrincDiv}(F) \in Cl(F)$ the divisor class of D. Since principal divisors have degree 0, we can write $\deg[D] := \deg(D)$. The group of degree zero divisor classes is $Cl^0(F) := \{[A] \in Cl(F) \mid \deg[A] = 0\}$ [42, Definition 5.1.2]. We call the order of $Cl^0(F)$ the divisor class number of F and denote it by h_F.

Let D be a divisor of F. Then we define the Riemann-Roch space $L(D)$ as follows.

Definition 2.17. [42, Definition 1.4.4] For a divisor $D \in Div(F)$ we define the *Riemann-Roch space* associated to D by

$$L(D) := \{\alpha \in F^\times | \mathrm{div}(\alpha) \geq -D\} \cup \{0\}$$

$L(D)$ is a finite dimensional vector space over k [42, Proposition 1.4.9]. For example, the Riemann-Roch space of the zero divisor is $L(0) = k$, and if $D < 0$, then $L(D) = \{0\}$ [42, Lemma 1.4.7]. The dimension of $L(D)$ is denoted by $\dim L(D)$.

Lemma 2.18. *[42, Corollary 1.4.12 (c)] For a divisor D of degree 0 the following are equivalent.*

(a) D is principal

(b) $\dim L(D) \geq 1$

(c) $\dim L(D) = 1$

With these concepts, we define the genus g of F as in [42, Definition 1.4.15].

$$g := \max\{\deg D - \dim L(D) + 1 \mid D \in Div(F)\}.$$

The genus of F is independent of k and the choice of an element $x \in F$ transcendental over k. When F is a global function field, g can be computed using the formula provided in [4, Corollary 3.5].

Minima and reduced ideals

In this subsection, we define the minima of an ideal and reduced ideals I, and present a bound on the degree of $\mathrm{div}(I)$ for a reduced ideal I. The material in this section is based on [15] and [18], and will be used mainly in Section 2.3 and Section 2.4.

Let F be a global function field. We define the minima of an ideal I and reduced ideals as follows. In order to define minima, we first define *boxes* in I as follows. A box in I can be defined with a real vector $v_\infty \in \mathbb{R}^{|\mathbb{P}_\infty(F)|}$, where $\mathbb{P}_\infty(F)$ is the set of all infinite places of F.

$$B(I, v_\infty) := \{\alpha \in I | \quad v_{P_i}(\alpha) \geq -(v_\infty)_i \text{ where } P_i \in \mathbb{P}_\infty(F) \text{ for } 1 \leq i \leq |\mathbb{P}_\infty(F)|\}.$$

For any element μ in I, we can define a box,

$$B\left(I, \left[v_{P_1}(\mu) \quad \ldots \quad v_{P_{|\mathbb{P}_\infty(F)|}}(\mu)\right]\right).$$

We call μ a minimum of I if the box above only contains constants and elements with the same values as μ at all $P \in \mathbb{P}_\infty(F)$. Here is a formal definition of minima of I.

Definition 2.19. [18, Definition 5.1] Let F be a global function field, and O_F be the finite maximal order of F.

- Let I be an ideal of O_F and $0 \neq \mu \in I$. The element μ is a *minimum of* I if for every $\alpha \in L(\mathrm{div}(I) - \sum_{P \in \mathbb{P}_\infty(F)} v_P(\mu)P)$, either $\alpha = 0$ or $v_P(\alpha) = v_P(\mu)$ for all $P \in \mathbb{P}_\infty(F)$.
- An ideal I is *reduced* if 1 is a minimum of I.
- Denote the set of all reduced ideals of O_F in the ideal class of I in $Cl(O_F)$ by $Red(I)$.

Let I be an ideal and μ a minimum of I. Due to the definition, we have the following. For any unit $\epsilon \in O_F^\times$ of O_F, $\mu\epsilon$ is also a minimum of I. Thus, the following map is well-defined

and bijective [18, p.2339].

$$\{\mu \mid \mu \text{ is a minimum of } I\}/O_F^\times \longrightarrow Red(I), \quad \mu O_F^\times \mapsto \frac{1}{\mu} I.$$

In fact, when I is an ideal of O_F with a minimum μ, $\frac{1}{\mu}I$ is a reduced ideal [18, p. 2338].

In addition, we have a useful bound on the degree of $\mathrm{div}(I_0)$ for a reduced ideal I_0. Combining [18, Remark 6.2(a)] and [18, Proposition 8.1], we have a bound on the degree of $\mathrm{div}(I_0)$,

$$0 \leq \deg \mathrm{div}(I_0) \leq g, \tag{2.6}$$

where g is the genus of F. This bound will be used for complexity analysis in Section 2.3.

2.1.3 Norm equations over global function fields

In this section, we first present the definition of norm equations over global function fields, and describe what it means to solve them.

A norm equation over a global function field F is defined as follows.

Definition 2.20. A *norm equation over* F is an equation of the form

$$\mathrm{Norm}_{F/k(x)}(\alpha) = c, \tag{2.7}$$

where $\alpha \in O_F$ and $c \in k(x) \setminus \{0\}$.

Typically, given c, we solve equation (2.7) by finding α in O_F that satisfies the norm equation $\mathrm{Norm}_{F/k(x)}(\alpha) = c$ up to a non-zero constant. In other words, we search for all $\alpha \in O_F$ such that $\mathrm{Norm}_{F/k(x)}(\alpha) = \zeta c$ for some constant $\zeta \in k^\times$. In addition, since we search for solutions in O_F, we only consider $c \in k[x] \setminus \{0\}$, because when $c \in k(x) \setminus k[x]$, solutions of (2.7) are not in O_F.

In order to obtain a finite solution space, we consider solutions up to units of O_F. Since we solve the norm equation up to a constant, if there exists one solution in O_F, that means

there are infinitely many solutions in O_F if the unit rank r is at least one. In particular, for a solution α of (2.7), and a system of fundamental units $\{\varepsilon_i | 1 \leq i \leq r\}$, any element of the form $\alpha \prod_{i=1}^{r} \varepsilon_i^{x_i}$ with arbitrary integers x_i is also a solution of the same norm equation (2.7) in O_F. Thus, our goal is to find all solutions of (2.7) up to units only, instead of finding all solutions. We define such elements as below.

Definition 2.21. Two elements α_1 and α_2 in O_F are *associate* if there exists a unit u of O_F that satisfies $\alpha_1 = u\alpha_2$. Otherwise, they are non-associate.

With this definition, we call those solutions that are differed by units by non-associate solutions.

From Definition 2.21, it is easy to show that for any elements α_1 and α_2 in O_F, they are associate if and only if for any finite place $P \in \mathbb{P}_0(F)$, $v_P(\alpha_1) = v_P(\alpha_2)$. This implies

$$v_P(\alpha_1) - v_P(\alpha_2) = 0, \tag{2.8}$$

for all $P \in \mathbb{P}_0(F)$. With this equality, we test if two elements are associate using the following algorithm.

Algorithm 1 IsAssociate

Input: α_1, $\alpha_2 \in O_F$.
Output: true if α_1 and α_2 are associate, false otherwise.

1: $S \longleftarrow \{P \in \mathbb{P}_0(F) \mid v_P(\alpha_1) \neq 0 \text{ or } v_P(\alpha_2) \neq 0\}$
2: **if** $v_P(\alpha_1) - v_P(\alpha_2) = 0$ for all $P \in S$ **then**
3: **return** true (associate)
4: **else**
5: **return** false (non-associate)
6: **end if**

2.2 Lattices and normed spaces

In this section, we first define lattices, normed spaces, and reduced bases. Then we describe how a function field F and a finite maximal order O_F can be viewed as a normed space and

a lattice, respectively. The material of this part of this section is from [5]. Later in Section 2.2.2, we discuss S-unit lattices of F. The material of Section 2.2.2 is mostly based on [24] and [34].

In order to define lattices and normed spaces, we first need to define absolute values on a ring R and norm (or length) functions on an R-module with an absolute value. Definition 2.23 is slightly revised from [13, p.14].

Definition 2.22. [13, p.1] Let R be a non-trivial ring. An *absolute value on* R is defined to be a function $|\ |: R \to \mathbb{R}$ such that the following hold.

(1) For any $\lambda \in R$, $|\lambda| \geq 0$, with equality if and only if $\lambda = 0$.

(2) For any $\lambda_1, \lambda_2 \in R$, $|\lambda_1 + \lambda_2| \leq |\lambda_1| + |\lambda_2|$.

(3) For any $\lambda_1, \lambda_2 \in R$, $|\lambda_1 \cdot \lambda_2| = |\lambda_1| \cdot |\lambda_2|$.

Definition 2.23. Let R be a ring with an absolute value $|\ |$. A *norm (or length) function* on an R-module L is a map $\|\ \| : L \longrightarrow \mathbb{R}$ such that

(1) $\|\alpha\| \geq 0$ with equality if and only if $\alpha = 0$.

(2) $\|\alpha + \beta\| \leq \|\alpha\| + \|\beta\|$, for all $\alpha, \beta \in L$

(3) $\|\lambda\alpha\| = |\lambda|\,\|\alpha\|$, for all $\lambda \in R$, $\alpha \in L$

Since we defined element norm earlier in this chapter, we will call the map in Definition 2.23 a length function in this chapter.

For example, the Euclidean norm is an absolute value on $\mathbb{Z}$ and also a length function on $\mathbb{Z}$-modules. In our case, R is either $k[x]$ or O_∞. In R, a map $|\ |: R \longrightarrow \mathbb{R}$ such that for $\lambda \in R$

$$|\lambda| = q^{v_\infty(\lambda)} \tag{2.9}$$

with v_∞ defined in (2.2) is an absolute value. The following two functions $\|\ \|_\infty$ and $\|\ \|$ are length functions on O_F, where the ramification index of $P \in \mathbb{P}_\infty(F)$ is denoted by e_P.

$\| \ \|_\infty : O_F \longrightarrow \mathbb{R} \cup \{-\infty\}$; for $\alpha \in O_F$, defined by

$$\|\alpha\|_\infty = \max_{P \in \mathbb{P}_\infty(F)} q^{-v_P(\alpha)/e_P}. \tag{2.10}$$

$\| \| : O_F \longrightarrow \mathbb{R} \cup \{-\infty\}$; for $\alpha \in O_F$, defined by

$$\|\alpha\| = \min_{P \in \mathbb{P}_\infty(F)} q^{-v_P(\alpha)/e_P}. \tag{2.11}$$

With length functions, we define lattices and normed spaces as follows.

Definition 2.24. [3, Definition 2.1.6] Let R be a ring and F_R be the field of fractions of R.

- Let L be a finitely generated R-module and $\| \|$ be a length function on L. The pair $(L, \| \|)$ is said to be a *lattice* over R if $\dim_k\{x \in L \mid \|x\| \leq r\} < \infty$ for all $r \in \mathbb{R}$.
- A *normed space over* F_R is a pair $(E, \| \|)$, where E is a finite dimensional F_R-vector space equipped with a length function $\| \|$, admitting a finitely generated R-submodule $L \subset E$ of full rank such that $(L, \| \|)$ is a lattice.

A basis of $(L, \| \|)$ refers to an R-basis of L and a basis of $(E, \| \|)$ refers to an F_R-basis of E.

For example, $\mathbb{Z}$ is a lattice with Euclidean norm, and $\mathbb{Z}$-modules are normed spaces with Euclidean norm. In our setting, $(k[x], | \ |)$ is a lattice and $(k(x), | \ |)$ is a normed space where $| \ |$ is defined as in (2.9). Moreover, $(O_F, \| \|_\infty)$ and $(O_F, \| \|)$ are lattices and $(F, \| \|_\infty)$ and $(F, \| \|)$ are normed spaces with the length functions defined in (2.10) and (2.11), respectively. We can also define lattices over the integer ring $\mathbb{Z}$ that are associated to F and are related to the unit group of O_F and S-unit groups. They will be discussed later in Section 2.2.2.

2.2.1 Additive norm functions

In this subsection, we define additive norm functions, which are derived by taking a logarithm of length functions in Definition 2.23. Using additive norms simplifies some computations in

Section 3.1 related to the length of $\alpha \in F$ as given in (2.10).

We consider taking the logarithm of an absolute value on R and a norm on an R-module. Since absolute values and norms defined in Definition 2.22 and Definition 2.23 are multiplicative, taking the logarithm makes them additive.

Definition 2.25. [3, Definition 2.1.1] Let R be a ring with an absolute value $|\ |$. An *additive norm (or length) function* on an R-module L is a map

$$\|\ \|_+ : L \longrightarrow \mathbb{R} \cup \{-\infty\}$$

satisfying the following conditions:

(1) $\|\alpha\|_+ = -\infty$ if and only if $\alpha = 0$.

(2) $\|\alpha + \beta\|_+ \leq \max\{\|\alpha\|_+, \|\beta\|_+\}$, for all $\alpha, \beta \in L$

(3) $\|\lambda\alpha\|_+ = \log|\lambda| + \|\alpha\|_+$, for all $\lambda \in R$, $\alpha \in L$

For example, with the length functions defined in (2.10) and (2.11), we define the respective associated additive norms as follows.

$$\|\alpha\|_{\infty,+} = \log_q \|\alpha\|_\infty \quad \text{and} \quad \|\alpha\|_+ = \log_q \|\alpha\|\,. \tag{2.12}$$

We often call the first additive norm $\|\alpha\|_{\infty,+}$ defined above the maximum additive norm.

With additive norms, we can define lattices and normed spaces the same way as in Definition 2.24. $(O_F, \|\ \|_+)$ is a lattice and $(F, \|\ \|_+)$ is a normed space [5, Theorem 3.1].

Now we define a reduced basis of a lattice or a normed space with an additive norm. To define it, we define the notion of reducedness. We abuse our notation in the definition below that L is a lattice or a normed space. Note that in the definition below, when L is a lattice, R is a ring, and when L is a normed space, R is a field of fractions of a ring.

Definition 2.26. [5, Definition 1.7] Let $(L, \| \ \|_+)$ be a *lattice* or a *normed space* with an additive norm $\| \ \|_+$ over R and $\mathcal{B} = \{\omega_1, \ldots, \omega_m\}$ be a subset of $L \setminus \{0\}$. We say that $\mathcal{B}$ is reduced if for all $\lambda_1, \ldots, \lambda_m \in R$,

$$\|\lambda_1\omega_1 + \cdots + \lambda_m\omega_m\|_+ = \max_{1\le i\le m} \|\lambda_i\omega_i\|_+ .$$

An R-basis of L is called a reduced basis if it is reduced and ordered by non-decreasing length.

In fact, every lattice with an additive norm has a reduced basis [5, Corollary 1.13]. In this thesis, a reduced basis of $(O_F, \| \ \|_+)$ will be referred to as a reduced basis of O_F. It can be computed by [5, Algorithm 1]. In addition, we can bound the length of elements in a reduced basis of O_F. Let $\mathcal{B} = \{\omega_1, \ldots \omega_n\}$ be a reduced basis of O_F. Then we have

$$\|\omega_n\|_+ \le \|\omega_n\|_{\infty,+} \le \left\lceil \frac{2g-1}{n} \right\rceil + 1. \tag{2.13}$$

The second inequality holds by [44, Theorem 5.4.1].

Let $\alpha = \sum_{i=1}^n \lambda_i\omega_i \in O_F$. Then we can bound the degrees of λ_i with $\|\alpha\|_{\infty,+}$ and $\|\omega_i\|_{\infty,+}$ as in the following lemma.

Lemma 2.27. *Let $\alpha = \sum_{i=1}^n \lambda_i\omega_i$ in O_F where $\mathcal{B} = \{\omega_i | 1 \le i \le n\}$ is a reduced basis of O_F, and $\lambda_i \in k[x]$. Then for $1 \le i \le n$,*

$$\deg \lambda_i \le \|\alpha\|_{\infty,+} - \|\omega_i\|_{\infty,+}.$$

Proof. Since $\mathcal{B}$ is a reduced basis, we know that

$$\begin{aligned} \|\alpha\|_{\infty,+} &= \left\| \sum_{i=1}^n \lambda_i\omega_i \right\|_{\infty,+} = \max_{1\le i\le n} \left\{ \|\lambda_i\omega_i\|_{\infty,+} \right\} \\ &= \max_{1\le i\le n} \{\log_q|\lambda_i| + \|\omega_i\|_{\infty,+}\} = \max_{1\le i\le n} \{\deg \lambda_i + \|\omega_i\|_{\infty,+}\}. \end{aligned} \tag{2.14}$$

That means, for each i for $1 \leq i \leq n$, we have

$$\|\alpha\|_{\infty,+} \geq \deg \lambda_i + \|\omega_i\|_{\infty,+}$$

This inequality leads to the result of this lemma directly. □

2.2.2 S-unit lattice

In this section, we present definitions and useful properties related to S-units of F, the S-unit lattice, and the S-unit valuation lattice. S-units are a generalization of units of O_F. The materials in this section is from [15] and [34, Chapter 14].

An S-unit lattice is a lattice over the integer ring $\mathbb{Z}$. Before discussing S-unit lattices, we set some definitions and notation we use in $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ and lattices over them. Let m be a positive integer. Then for a vector $v = \begin{bmatrix} v_1 & \dots & v_m \end{bmatrix}^T \in \mathbb{R}^m$, we have different norms. First of all, in $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$, we use $\| \ \|$ as the Euclidean norm. We also have the maximum norm, which is denoted by $\| \ \|_\infty$. Let $v \in \mathbb{R}^m$, then we use $(v)_i$ as i-th entry of v, and the maximum norm $\|v\|_\infty$ of v is defined by

$$\|v\|_\infty := \max_{1 \leq i \leq m} |(v)_i|.$$

For an $m_1 \times m_2$ matrix M over $\mathbb{R}$, we use $(M)_i$ as i-th row of M, $(M)_{\cdot,j}$ as j-th column of M, and $(M)_{i,j}$ as i,j-th entry of M. The maximum norm $\|M\|_\infty$ of M is defined by

$$\|M\|_\infty := \max_{\substack{1 \leq i \leq m_1 \\ 1 \leq j \leq m_2}} |(M)_{i,j}|.$$

Now we define the ring of S-integers of F and S-units for some set S as in [34, p.241-242]. Let S be a finite set of places of F including all infinite places of F. Then we define the ring

of S-integers as below.

$$O_S := \{\alpha \in F \mid v_P(\alpha) \geq 0, \quad \forall P \notin S\}.$$

Note that $O_S \subset O_F$ is a ring [34, p.241]. We denote the set of nonzero fractional ideals, the set of principal ideal, the ideal class group, and the ideal class number of O_S by $\mathrm{Id}(O_S)$, $\mathrm{PrincId}(O_S)$, $Cl(O_S)$, and h_{O_S}, respectively. Due to construction, $h_{O_S} \leq h_{O_F}$.

The S-units are the units of O_S. The S-unit group $O_S^\times$ is the unit group of O_S,

$$O_S^\times = \{\alpha \in F^\times \mid v_P(\alpha) = 0, \quad \forall P \notin S\}. \tag{2.15}$$

Let $a \in k$ be a nonzero constant. Since $v_P(a) = 0$ for all $P \in \mathbb{P}(F)$, a is in $O_S^\times$. In fact, $O_S^\times/k^\times$ is finitely generated free abelian group of rank $r_S = |S| - 1$ [34, Proposition 14.1]. A set $\{\epsilon_1, \ldots, \epsilon_{|S|-1}\}$ of S-units that generates $O_S^\times/k^\times$ is called a set of fundamental S-units. When $S = \mathbb{P}_\infty(F)$, the S-units, the S-unit group $O_S^\times$, the S-unit rank r_S, and a set of fundamental S-units are simply referred to as units, the unit group $O_F^\times$, the unit rank r, and a set of fundamental units, respectively.

We denote the subgroup of $Div(F)$ generated by S by $Div(S)$ and $Div^0(F) \cap Div(S)$ by $Div^0(S)$, and define $\mathrm{PrincDiv}(S)$ as the set of all principal divisors in $Div(S)$, $\mathrm{PrincDiv}(S) := Div(S) \cap \mathrm{PrincDiv}(F)$. Then $\mathrm{PrincDiv}(F)$ is a subgroup of $\mathrm{PrincDiv}(F)$ and $Div^0(S)$ [34, p.242]. We denote the subgroup of $Div(F)$ generated by the places not in S by $Div_S(F)$, and define $\mathrm{PrincDiv}_S(F) := Div_S(F) \cap \mathrm{PrincDiv}(F)$. We define the S-class group as

$$Cl_S(F) := Div_S(F)/\mathrm{PrincDiv}_S(F).$$

Let C be the cyclic group of order d_S/d_F, where d_S is the great common divisor of $\{\deg P | P \in S\}$, and d_F is the great common divisor of $\{\deg P | P \in \mathbb{P}(F)\}$. Then we have

the following exact sequence [34, Lemma 14.1 (b)].

$$(0) \to Div(S)/\text{PrincDiv}(S) \to Cl^0(F) \to Cl_S(F) \to C \to (0). \tag{2.16}$$

$Cl_S(F)$ is isomorphic to the ideal class group $Cl(O_S)$ of O_S [34, Theorem 14.5]. Thus, $|Cl_S(F)| = |Cl(O_S)| = h_{O_S}$.

We consider two kinds of lattices over $\mathbb{Z}$ that are related to S-units. One is the S-unit lattice, and the other is a lattice we call the S-unit valuation lattice. We define them as follows.

Let $S = \{P_1, \dots P_{|S|}\}$ be a finite set of places of F including all infinite places, $O_S^\times$ be the S-unit group, and $\{\epsilon_1, \dots, \epsilon_{|S|-1}\}$ be a set of fundamental S-units. The S-unit lattice Λ_S is defined as follows. Let Φ_S be a map from $F^\times$ to $\mathbb{Z}^{|S|}$,

$$\Phi_S : F^\times \longrightarrow \mathbb{Z}^{|S|}, \qquad \alpha \mapsto \left(-v_{P_1}(\alpha) \deg P_1, \ \dots, \ -v_{P_{|S|}}(\alpha) \deg P_{|S|}\right).$$

Denoting the image of $O_S^\times$ under Φ by $\Phi_S(O_S^\times)$, we have $\Lambda_S := (\Phi_S(O_S^\times), \| \ \|)$ is a lattice over $\mathbb{Z}$ where $\| \ \|$ is the Euclidean norm. When $S = \mathbb{P}_\infty(F)$, the image of the unit group $O_F^\times$ of O_F under the map $\Phi_{\mathbb{P}_\infty(F)}$ is the unit lattice $\Lambda := (\Phi_{\mathbb{P}_\infty(F)}(O_F^\times), \| \ \|)$ of F with Euclidean norm $\| \ \|$. Let $\{\epsilon_1, \dots, \epsilon_{|S|-1}\}$ be a set of fundamental S-units. Then $\{\Phi_S(\epsilon_1), \dots, \Phi_S(\epsilon_{|S|-1})\}$ is a basis of Λ_S.

Let $\mathcal{B}$ be a basis of Λ_S. Then we can form the $(|S|-1) \times |S|$ matrix M_{Λ_S} whose rows consist of elements of $\mathcal{B}$. The determinant of any $(|S|-1) \times (|S|-1)$ minor of M_{Λ_S} is called the S-regulator R_S [34, Lemma 14.3]. R_S is does not depend on the choice of $\mathcal{B}$ or the choice of minor [34, p.245].

When $S = \mathbb{P}_\infty(F)$, we call R_S the regulator of O_F, and denote it by R_F. By the Hasse-Weil Theorem in [42, Theorem 5.1.15 and Theorem 5.2.1], we have bounds on R_F,

$$(\sqrt{q}-1)^{2g} \leq h_{O_F} R_F \leq (\sqrt{q}+1)^{2g}. \tag{2.17}$$

Now we consider the S-unit valuation lattice. The S-unit valuation lattice is defined using the map ϕ_S from $F^\times$ to $\mathbb{Z}^{|S|}$,

$$\phi_S : F^\times \longrightarrow \mathbb{Z}^{|S|}, \qquad \alpha \mapsto \left(-v_{P_1}(\alpha), \ldots, -v_{P_{|S|}}(\alpha)\right). \tag{2.18}$$

Let $\phi_S(O_S^\times)$ be the image of $O_S^\times$ under the map ϕ_S. Then $\Lambda'_S := (\phi_S(O_S^\times), \| \|)$ is a lattice with the Euclidean norm $\| \|$ [15, p.341], which we call the S-unit valuation lattice. The set

$$\left\{\phi_S(\epsilon_1), \ldots, \phi_S(\epsilon_{|S|-1})\right\} \tag{2.19}$$

is a basis of Λ'_S, where $\{\epsilon_i |\ 1 \leq i \leq |S| - 1\}$ is a set of fundamental S-units.

Similar to the S-regulator, we define R'_S as follows. Let $\mathcal{B}'$ be a basis of Λ'_S. Then we can form the $(|S|-1) \times |S|$ matrix $M_{\Lambda'_S}$ whose rows consist of elements of $\mathcal{B}'$. Let $(M_{\Lambda'_S})_{(i,j)}$ be the (i,j) minor of $M_{\Lambda'_S}$. Then we define

$$R'_S = \det(M_{\Lambda'_S})_{(i,j)}. \tag{2.20}$$

Similar to R_S, R'_S does not depend on the choice of $\mathcal{B}'$ nor i and j. Due to the construction, $R'_S \leq R_S$. When $S = \mathbb{P}_\infty(F)$, we denote R'_S by R'_F.

Lemma 2.28. *Let S be a subset of $\mathbb{P}(F)$ including all the infinite places of F. Then*

$$R'_S \leq h_{O_F} R_F.$$

Proof. Following the proof of [34, Lemma 14.3] for R'_S, we have

$$R'_S = \frac{[Div^0(S) : \mathrm{PrincDiv}(S)]}{d_S},$$

where d_S is the greatest common divisor of $\{\deg P | P \in S\}$. Using the equality

$$|Cl^0(F)|d_S = |Cl_S(F)|[Div^0(S) : \text{PrincDiv}(S)]$$

from [34, p.247], we have $R'_S = \frac{|Cl^0(F)|}{|Cl_S(F)|}$. Similarly, $R'_F = \frac{|Cl^0(F)|}{|Cl_{\mathbb{P}_\infty(F)}(F)|}$. Thus,

$$R'_S = \frac{|Cl_{\mathbb{P}_\infty(F)}|}{|Cl_S(F)|} R'_F.$$

In addition, from [34, Theorem 14.5], $|Cl_S(F)| = |Cl(O_S)| = h_{O_S}$ and $|Cl_{\mathbb{P}_\infty(F)}(F)| = |Cl(O_F)| = h_{O_F}$. Thus, R'_S satisfies the following bound.

$$\begin{aligned} R'_S &\leq \frac{h_{O_F}}{h_{O_S}} R'_F \\ &\leq h_{O_F} R'_F \\ &\leq h_{O_F} R_F. \end{aligned}$$

The second inequality is because h_{O_F} and h_{O_S} are positive integers. □

When F is a global function field, a basis of Λ'_S can be computed as in [24, p.21-22] as an application of [25, Algorithm 5.5]. The following is a short summary of the process described in [24, p.21-22]. Consider the residue class homomorphism

$$\psi_S : Div_S(F) \longrightarrow Cl(F).$$

The kernel of ψ_S is $\text{PrincDiv}_S(F)$. Note that we have isomorphisms $\mathbb{Z}^{|S|} \cong Div_S(F)$ and $Cl(F) \cong \mathbb{Z}/c_1\mathbb{Z} \times \cdots \times \mathbb{Z}/c_m\mathbb{Z} \times \mathbb{Z}$ for some $c_1, \ldots, c_m \in \mathbb{Z}$ and $c_i \mid c_j$ for $i \leq j$ and $1 \leq m \leq 2g$. The integers $c_1, \ldots, c_m$ can be determined using Hess's relation search algorithm

[25, Algorithm 5.5.]. The maps between $\mathbb{Z}^{|S|}$ and $Div_S(F)$ are defined by

$$\psi_1 : \mathbb{Z}^{|S|} \longrightarrow Div_S(F), \qquad (v_1, \ldots, v_{|S|}) \mapsto \sum_{i=1}^{|S|} v_i P_i,$$

$$\psi_2 : Div_S(F) \longrightarrow \mathbb{Z}^{|S|}, \qquad D \mapsto \left(v_{P_1}(D), \ldots, v_{P_{|S|}}(D)\right).$$

With those isomorphisms, a new homomorphism ψ'_S can be defined as below,

$$\psi'_S : \mathbb{Z}^{|S|} \longrightarrow \mathbb{Z}/c_1\mathbb{Z} \times \cdots \times \mathbb{Z}/c_m\mathbb{Z} \times \mathbb{Z}. \tag{2.21}$$

Then the kernel of ψ'_S is isomorphic to $\mathrm{PrincDiv}_S(F)$, thus a basis of the kernel of ψ'_S is a basis of the S-unit valuation lattice.

Algorithm 2 performs the above process and LLL reduction of the basis. Note that the output of the algorithm is an $(|S|-1) \times |S|$ matrix M_S, whose rows form a basis of the S-unit valuation lattice. In other words, each row of M_S is an integer vector defined by the values of a fundamental S-unit at the places in S, as in the set in (2.19). We refer to the output of the algorithm as an S-unit value matrix.

Algorithm 2 SValMat

Input: a finite set $S = \{P_1, \ldots, P_{|S|}\}$ of places including all infinite places of F
Output: An S-unit value matrix matrix $M_S \in \mathbb{Z}^{(|S|-1)\times|S|}$.

1: Compute the structure of $Cl(F)$ using [25, Algorithm 5.5]
2: $\psi'_S \longleftarrow$ the map defined in (2.21)
3: $\{v_1, \ldots, v_{|S|}\} \longleftarrow$ Generators of the kernel of ψ'_S
4: $M_0 \longleftarrow (|S|-1) \times |S|$ matrix whose i-th row is v_i for $1 \le i \le |S|-1$
5: $M_S \longleftarrow$ LLL reduction on M_0
6: **return** M_S

Using the same algorithm with input $\mathbb{P}_\infty(F)$, we can also compute a basis of the unit valuation lattice, i.e. a matrix of valuations of a set of fundamental units. In addition, later in this thesis, we split M_S into two parts, $M_{S,0}$ and $M_{S,\infty}$, where $M_{S,0}$ is the matrix of all columns of values at the finite places of F and $M_{S,\infty}$ is the matrix of the rest of M_S, which

is the matrix of columns of the values at the infinite places of F.

Let M_S be an output of Algorithm 2 with a set S of places of F. We can bound $\|M_S\|_\infty$ as follows.

Proposition 2.29. *Let S be a set of places of F, including all infinite places of F, M_S be the output of Algorithm 2 with S, and R'_S be as in (2.20). Then*

$$\|M_S\|_\infty \leq 2^{(|S|-1)(|S|-2)/4} R'_S$$

Proof. From M_S, we have an LLL-reduced basis $\{(M_S)_1, \ldots, (M_S)_{|S|-1}\}$ of Λ'_S. Then by [30, Proposition 1.6], we have

$$\prod_{j=1}^{|S|-1} \|(M_S)_j\| \leq 2^{(|S|-1)(|S|-2)/4} \det \Lambda'_S.$$

Since $(M_S)_j$ are nonzero integer vectors, we have

$$\|(M_S)_j\| \leq \prod_{j=1}^{|S|-1} \|(M_S)_j\|$$

for any $1 \leq j \leq |S|-1$. Finally, since $\|M_S\|_\infty \leq \|(M_S)_j\|$ for any $1 \leq j \leq |S|-1$, we have the result. □

Corollary 2.30. *Let S be a set of places of F, including all infinite places of F, and $\mathcal{B}_S = \{b_1, \ldots, b_{|S|-1}\}$ be a LLL-reduced basis of the S-unit lattice Λ_S. Then*

$$\max_{1 \leq i \leq |S|-1} \|b_i\|_\infty \leq 2^{(|S|-1)(|S|-2)/4} R_S.$$

2.3 Computational costs of fundamental arithmetic in function fields

In this section, we analyze the computational cost of fundamental arithmetic in function fields. We start with the costs of multiplications and exponentiations of polynomials, rational functions, and elements of F. Then we describe the costs of computing with HNF representations of ideals of O_F.

For every complexity analysis in this section, we suppose that we have $k = \mathbb{F}_q$ fixed, but the extension degree n, the genus g and the degree of c of Equation (2.7) can vary, unless specified otherwise. All computational costs are expressed in terms of the number of bit operations. We use the fast multiplication technique for finite field elements and polynomials in [23] for our complexity analysis. Using that, an operation in k takes $O(\log q \log\log q)$ bit operations. For simplicity, we will generally estimate any power of $\log x$ (for any quantity x) by x^{ε}. For example, we simplify the number of bit operations needed for an operation in k as $O(\log q \log\log q) = O\left((\log q)^{1+\varepsilon}\right) = O(q^{\varepsilon})$. We also assume that a fractional ideal is given in HNF representation described in Definition 2.16.

We start with the cost of multiplication of polynomials of bounded degrees, then generalize it to the cost of multiplications of rational functions and elements in F of bounded height.

Let $\mathcal{M}(d)$ be the number of bit operations to multiply two polynomials in $k[x]$ of degrees bounded by d. By computing naively, $\mathcal{M}(d) = O(d^2 q^{\varepsilon})$. Using the fast multiplication technique in [23], $\mathcal{M}(d) = O(d^{1+\varepsilon} q^{\varepsilon})$ bit operations.

In order to analyze the cost of multiplying rational functions and elements of F of bounded height, we recall Definitions 2.2 and 2.3. The height will also be used to estimate the sizes of elements of $k(x)$ and F later in this chapter. We measure the sizes in the number of bits required to represent the elements. With the heights of $\lambda \in k(x)$ and $\alpha \in O_F$, we can bound the number of elements in k required to represent λ and α, and thus the number of bits

required. The number of elements in k required to represent λ is bounded by $2\mathbf{h}(\lambda)$, and the number of elements in k required to represent α with respect to a reduced basis $\mathcal{B}$ is bounded by $2n\mathbf{h}_{\mathcal{B}}(\alpha)$, and we suppose that representing an element of k requires $O(\log q)$ bits.

Clearly, multiplying two rational functions in $k(x)$ of heights bounded by d takes $O(\mathcal{M}(d))$. The following lemma provides the cost of multiplying m rational functions.

Lemma 2.31. *Let m be an positive integer, $\lambda_i(x) \in k(x)$ for $1 \leq i \leq m$, and $d = \max_{1\leq i\leq m}\{\mathbf{h}(\lambda_i(x))\}$. Then the number of bit operations to compute $\prod_{i=1}^{m}\lambda_i(x)$ is bounded by*

$$2(m-1)\mathcal{M}((m-1)d).$$

Proof. Computing $\prod_{i=1}^{m}\lambda_i(x)$ takes $2(m-1)$ multiplications of rational functions. The most expensive multiplication among them is $(\prod_{i=1}^{m-1}\lambda_i(x))\lambda_m(x)$, which costs $2\mathcal{M}((m-1)d)$ bit operations. Thus, the total number of bit operations is bounded as stated in this lemma. □

Next, we consider multiplications of elements in F. We assume that the elements are represented with respect to a reduced basis $\mathcal{B} = \{\omega_i | 1 \leq i \leq n\}$ of O_F, for which we precompute $\omega_i\omega_j$ for $1 \leq i, j \leq n$. To be specific, we precompute all $a_{i,j,k} \in k[x]$ such that $\omega_i\omega_j = \sum_{k=1}^{n} a_{i,j,k}\omega_k$ for $1 \leq i, j \leq n$. We recall [44, Proposition 5.4.4.] that $\mathbf{h}(a_{i,j,k})$ are bounded as below.

$$\mathbf{h}(a_{i,j,k}) = \deg(a_{i,j,k}) \leq 2\left(\left\lceil\frac{2g-1}{n}\right\rceil + 1\right).$$

Note that $\lceil\cdot\rceil$ denotes the smallest integer that is greater than or equal to the number inside. Using this bound, we bound the cost of multiplying elements of F represented with respect to $\mathcal{B}$.

Lemma 2.32. *Let $\mathcal{B} = \{\omega_i | 1 \leq i \leq n\}$ be a reduced basis of O_F, $m > 1$ be a positive integer,*

$\alpha_i = \sum_{j=1}^{n} \lambda_{i,j}\omega_j \in F$ *where* $\lambda_{i,j} \in k(x)$ *for* $1 \le i \le m$, *and*

$$d = \max\left\{ \max_{1\le i\le m} \{\mathbf{h}_{\mathcal{B}}(\alpha_i)\}, 2\left(\left\lceil \frac{2g-1}{n} \right\rceil + 1\right)\right\}.$$

Then computing $\prod_{i=1}^{m} \alpha_i$ *takes up to*

$$4(m-1)n^3\mathcal{M}((m-1)d)$$

bit operations.

Proof. First we consider the cost of computing $\alpha_1\alpha_2$ and generalize the result to $\prod_{i=1}^{m} \alpha_i$. We have

$$\begin{aligned}
\alpha_1\alpha_2 &= \left(\sum_{j=1}^{n} \lambda_{1,j}\omega_j\right)\left(\sum_{k=1}^{n} \lambda_{2,k}\omega_k\right) \\
&= \sum_{j=1}^{n}\sum_{k=1}^{n} \lambda_{1,j}\omega_j\lambda_{2,k}\omega_k \\
&= \sum_{j=1}^{n}\sum_{k=1}^{n} \lambda_{1,j}\lambda_{2,k}\left(\sum_{m=1}^{n} a_{j,k,m}\omega_m\right) \\
&= \sum_{m=1}^{n} \omega_m \sum_{j=1}^{n}\sum_{k=1}^{n} \lambda_{1,j}\lambda_{2,k}a_{j,k,m}.
\end{aligned}$$

The third equality is from the precomputed $\omega_j\omega_k$. The cost of computing each $\lambda_{1,j}\lambda_{2,k}a_{j,k,m}$ is $4\mathcal{M}(2d)$ by Lemma 2.31, and we perform n^3 such multiplications. Thus, computing $\alpha_1\alpha_2$ takes $4n^3\mathcal{M}(2d))$ bit operations.

Computing $\prod_{i=1}^{m} \alpha_i$ takes $m-1$ such multiplications of elements in F. Note that $\mathbf{h}_{\mathcal{B}}(\prod_{i=1}^{m-1} \alpha_i) \le (m-1)d$. Thus, the cost of computing $\prod_{i=1}^{m} \alpha_i$ is as stated. □

With the above lemmas, we estimate the costs of computing the m-th power of a rational function, and of an element of F for a positive integer m. We adopt the binary exponentiation for these computations.

Lemma 2.33. *Let $\lambda \in k(x)$, $d = \mathbf{h}(\lambda)$, and m be a positive integer. Then computing λ^m by binary exponentiation takes up to*

$$4\mathcal{M}((|m|-1)d)\log|m|$$

bit operations.

Proof. Using binary exponentiation, it takes up to $\log|m|$ squarings and $\log|m|$ multiplications of rational functions of height less than $(|m|-1)d$. □

Lemma 2.34. *Let $\mathcal{B} = \{\omega_i | 1 \leq i \leq n\}$ be a reduced basis of O_F, $\alpha = \sum_{i=1}^{n} \lambda_i \omega_i \in F$, $d = \max\{h_{\mathcal{B}}(\alpha), 2\left(\lceil \frac{2g-1}{n} \rceil + 1\right)\}$, and $m \in \mathbb{Z} \setminus \{0\}$. Then computing α^m by binary exponentiation takes up to*

$$8n^3\mathcal{M}(|m|d)\log|m|$$

bit operations.

Proof. Using binary exponentiation, it takes up to $\log|m|$ squarings and $\log|m|$ multiplications. Due to Lemma 2.32, each multiplication and squaring takes up to $4n^3\mathcal{M}(|m|d)$ bit operations. □

For a rational function $\lambda \in k(x)$, we have the following inequality on the sum of values of λ at the finite places of F and the height of λ. We will recall this lemma later in this chapter to analyze the complexity of computing compact representations and the size of compact representations.

Lemma 2.35. *Let $\lambda \in k(x)$ be given as*

$$\lambda = \frac{\prod_{i=1}^{m_1} \lambda_{1,i}}{\prod_{i=1}^{m_2} \lambda_{2,i}},$$

where $\lambda_{1,i}, \lambda_{2,i} \in k[x]$ *are irreducible and* $\prod_{i=1}^{m_1} \lambda_{1,i}$ *and* $\prod_{i=1}^{m_2} \lambda_{2,i}$ *are coprime. Then*

$$\sum_{P \in \mathbb{P}_0(F)} |v_P(\lambda)| \deg P \leq 2n\mathbf{h}(\lambda).$$

Proof. First we divide the sum into two parts, one for $\prod_{i=1}^{m_1} \lambda_{1,i}$ and the other for $\prod_{i=1}^{m_2} \lambda_{2,i}$:

$$\sum_{P \in \mathbb{P}_0(F)} |v_P(\lambda)| \deg P \leq \sum_{i=1}^{m_1} \sum_{P \in \mathbb{P}_0(F)} v_P(\lambda_{1,i}) \deg P + \sum_{i=1}^{m_2} \sum_{P \in \mathbb{P}_0(F)} v_P(\lambda_{2,i}) \deg P. \tag{2.22}$$

Since $\lambda_{1,i}, \lambda_{2,i}$ are irreducible, for $j \in \{1, 2\}$, we have

$$v_P(\lambda_{j,i}) = \begin{cases} 0 & \text{if } \mathfrak{p} \nmid \lambda_{1,i} O_F, \\ e_P & \text{if } \mathfrak{p} \mid \lambda_{1,i} O_F, \end{cases}$$

where $\mathfrak{p}$ is the prime ideal of O_F corresponding to the finite place $P \in \mathbb{P}_0(F)$ and e_P is the ramification index of P. Let f_P be the residue degree of P. With the above equality and Equation (2.3), we can rewrite the right hand side of (2.22) as below.

$$\begin{aligned} \sum_{i=1}^{m_1} \sum_{P \mid \lambda_{1,i}} e_P \deg P + \sum_{i=1}^{m_2} \sum_{P \mid \lambda_{2,i}} e_P \deg P &= \sum_{i=1}^{m_1} \sum_{P \mid \lambda_{1,i}} e_P f_P \deg \lambda_{1,i} + \sum_{i=1}^{m_2} \sum_{P \mid \lambda_{2,i}} e_P f_P \deg \lambda_{2,i} \\ &= \sum_{i=1}^{m_1} n \deg \lambda_{1,i} + \sum_{i=1}^{m_2} n \deg \lambda_{2,i} \leq 2n\mathbf{h}(\lambda). \end{aligned}$$

□

Now we consider computing the cost of computing the norm of an element of F. Let $\alpha \in F$. In order to compute the norm of α, we first compute the matrix $M_\alpha \in k(x)^{n \times n}$ that satisfies

$$\alpha \begin{bmatrix} \omega_1 & \dots & \omega_n \end{bmatrix} = \begin{bmatrix} \omega_1 & \dots & \omega_n \end{bmatrix} M_\alpha, \tag{2.23}$$

where $\{\omega_i | 1 \leq i \leq n\}$ is a reduced basis of O_F. Then we compute $\mathrm{Norm}_{F/k(x)}(\alpha) = \det(M_\alpha)$. Note that the cost of computing the determinant of a matrix in $k[x]^{n \times n}$ can be expressed

as a function of the cost of matrix multiplication by [29, Algorithm 2]. Computing the determinant of a matrix $M \in k[x]^{n\times n}$ takes

$$O(n^{\omega+\varepsilon}q^{\varepsilon}\lceil s\rceil) \tag{2.24}$$

bit operations, where s is the average column degree of M, defined as

$$\frac{\sum_{i=1}^{n}(\max_{1\le j\le n}\deg(M)_{i,j})}{n}, \tag{2.25}$$

$\lceil s\rceil$ denotes the smallest integer that is greater or equal to s, and ω is the matrix multiplication exponent [29, Proposition 3.3]. The best known bound on ω to date is $\omega < 2.37286$ [2]. The cost of computing the norm of an element is presented in the following lemma.

Lemma 2.36. *Let $\mathcal{B} = \{\omega_i | 1 \le i \le n\}$ be a reduced basis of O_F, $\alpha = \sum_{i=1}^{n}\lambda_i\omega_i \in F$ and $d_\alpha = \max\{\mathbf{h}_{\mathcal{B}}(\alpha), 2\left(\lceil\frac{2g-1}{n}\rceil + 1\right)\}$ Computing $Norm_{F/k(x)}(\alpha)$ takes*

$$O(n^3 d_\alpha^{1+\varepsilon}q^{\varepsilon})$$

bit operations.

Proof. Let M_α be defined as in 2.23. One can compute the entries of M_α as follows:

$$\begin{aligned}
\alpha\omega_j &= \left(\sum_{i=1}^{n}\lambda_i\omega_i\right)\omega_j \\
&= \sum_{i=1}^{n}\lambda_i\sum_{k=1}^{n}a_{i,j,k}\omega_k \\
&= \sum_{k=1}^{n}\omega_k\sum_{i=1}^{n}\lambda_i a_{i,j,k} \\
&= \sum_{k=1}^{n}(M_\alpha)_{j,k}\omega_k.
\end{aligned}$$

The second equality is from the precomputed quantities $\omega_i\omega_j$, and the last equality is obtained

by substituting $\sum_{i=1}^{n} \lambda_i a_{i,j,k}$ by $(M_\alpha)_{j,k}$. Computing each entry $(M_\alpha)_{j,k}$ costs $O(n\mathcal{M}(d_\alpha))$. Since M_α has n^2 entries, computing M_α costs $O\left(n^3\mathcal{M}\left(d_\alpha\right)\right)$. Since $\deg(M_\alpha)_{j,k} \leq 2d_\alpha$, we know the average column degree of M_α is also bounded by $2d_\alpha$, so computing the determinant of the matrix M_α costs $O\left(n^{\omega+\varepsilon}d_\alpha q^\varepsilon\right)$ bit operations by (2.24). Therefore, computing $\mathrm{Norm}_{F/k(x)}(\alpha)$ takes

$$O\left(n^{\omega+\varepsilon}d_\alpha q^\varepsilon + n^3\mathcal{M}\left(d_\alpha\right)\right) = O\left(n^3 d_\alpha^{1+\varepsilon} q^\varepsilon\right). \qquad \square$$

Let $(M_{I_1'}, d(I_1))$ and $(M_{I_2'}, d(I_2))$ be the HNF representations of two ideals I_1 and I_2. From [44, Theorem 5.2.2], we know computing $I_1 I_2$ costs $Cn^7 g^2 q^\varepsilon$ bit operations, where C is a constant.

We determine the costs of computing the norm of an ideal as follows.

Lemma 2.37. *Let $(M_{I'}, d(I))$ be the HNF representation of an ideal I of O_F. Then computing the norm of I costs*

$$O\left(\left(n^{2+\varepsilon}\deg(Norm(d(I)I)) + (n\deg d(I))^{1+\varepsilon}\right)q^\varepsilon\right)$$

bit operations.

Proof. Note that the degrees of any entries of $M_{I'}$ are bounded by $\deg(\mathrm{Norm}(d(I)I))$ [44, Proposition 5.1.17]. When given the HNF representation of I, the norm of I can be computed as $\mathrm{Norm}(I) = \frac{\det M_{I'}}{d(I)^n}$, as in (2.4). Since HNF matrix is upper-triangular, the cost of computing the determinant of $M_{I'}$ is the same with the cost of multiplying n polynomials of degree $\deg(\mathrm{Norm}(d(I)I))$ which is in Lemma 2.31. Thus, computing the determinant costs $O(n^{2+\varepsilon}\deg(\mathrm{Norm}(d(I)I))q^\varepsilon)$. The cost of computing $d(I)^n$ is $O((n\deg(d(I)))^{1+\varepsilon})q^\varepsilon$ as in Lemma 2.33, the result is obtained. $\square$

Lastly, the cost of computing the prime ideal factorization of an ideal I is dominated by

computing $\mathrm{Norm}(I)$. Using Berlekamp's algorithm for factoring polynomials, the cost is

$$O\left(((\deg \mathrm{Norm}(I))^{\omega} + (\deg \mathrm{Norm}(I))^{1+\varepsilon})q^{\varepsilon}\right), \tag{2.26}$$

given in [47, Theorem 14.32.]. When given I in the HNF representation $(M_{I'}, d(I))$, we know that $\deg \mathrm{Norm}(I) = \sum_{i=1}^{n} \deg(M_{I'})_{i,i}$. Thus, computing the prime ideal factorization of I takes

$$O\left(\left(\sum_{i=1}^{n} \deg(M_{I'})_{i,i}\right)^{\omega} q^{\varepsilon}\right) \tag{2.27}$$

bit operations.

The cost of testing associateness (Algorithm 1) is essentially computing values of given elements, the cost of which is dominated by the factorizations of the ideals generated by the input elements. Thus, the cost of Algorithm 1 is as in (2.26).

Lemma 2.38. *The cost of Algorithm 2 is subexponential in g, and polynomial in q, and n.*

Proof. The cost of Algorithm 2 is dominated by computing the divisor class group, which can be computed by Hess's relation searching algorithm described in [25, Algorithmus 5.5]. We consider the cost of computing the divisor class group when one of n, g, and q goes infinity and the others are fixed.

Let C_f defined as follows. Let $f(x,t) = t^n + a_1(x)t^{n-1} + \cdots + a_n(x) \in k[x][t]$ be the defining polynomial of F, where $a_i(x) \in k[x]$ for $1 \leq i \leq n$. We define

$$C_f = \max\{\lceil \deg a_i(x)/i \rceil | 1 \leq i \leq n\}. \tag{2.28}$$

The expected running time of computing the divisor class group is subexponential in g as $g \to \infty$ [25, Satz 5.23],

$$C_f^{O(1)} exp\left(\sqrt{g \log g}\left(5\sqrt{\frac{\log q}{6}} + o(1)\right)\right),$$

under the assumptions that q is fixed, n is bounded, and a smoothness assumption for divisors, which is described in [25, 4.19. Glattheitsannahme]. Secondly, the divisor class group can be computed in an expected time of

$$O(q^{2-2/(n-2)+\varepsilon})$$

when q goes infinity [14], which is polynomial in q. Finally, it is polynomial in n when n goes infinity because the representations of divisor classes and the cost of arithmetic in the divisor class group are polynomial in n [6, Theorem 3.11]. □

2.4 Compact representations

In this section, first, we present two ways of representing an element of a global function field F: a standard representation and a compact representation. Then in Section 2.4.1, we summarize algorithms to compute compact representations in [15]. In Section 2.4.2, we develop algorithms performing arithmetic operations using compact representations including multiplication, exponentiation, computing the norm, computing the value at a given place of F, and testing whether or not two elements are associate. Lastly, we analyze the complexity of all algorithms including the algorithm for computing compact representations, and the algorithms for arithmetic operations with compact representations in Section 2.4.3.

Let α be an element of a global function field F. Then we call the representation of α with respect to a basis $\mathcal{B} = \{\omega_i | 1 \leq i \leq n\}$ of the maximal order O_F of F the standard representation. The standard representation of α is

$$\alpha = \sum_{i=1}^{n} \lambda_i \omega_i, \tag{2.29}$$

where the coefficients $\lambda_i \in \mathbb{F}_q(x)$. The heights of λ_i can be very large even when the $\mathrm{Norm}_{F/k(x)}(\alpha)$ is small. For example from [35], the height of the fundamental unit ε_F of a

real quadratic function field $F_2 = \mathbb{F}_q(x)(\sqrt{D})$ with a square-free polynomial $D \in \mathbb{F}_q[x]$ is exponential in $\deg(D)$, even though its norm is in $\mathbb{F}_q^*$.

An alternative way of representing α is in a compact representation, which is in a power product form. Due to the multiplicative form, using compact representations is advantageous when conducting multiplicative operations, compared to the standard representation which has a linear combination form. In addition, the size of a compact representation is polynomial in the extension degree n of F, and $\mathbf{h}(\mathrm{Norm}_{F/k(x)}(\alpha))$, $\log q$, g, and $\log \|\mathrm{val}_\infty(\alpha)\|_\infty$. In this section, for $\alpha \in F$, we let $P_1, \ldots P_{r+1}$ by the infinite places of F where r is the unit rank of O_F, and denote the vector $\begin{bmatrix} v_{P_1}(\alpha) & \ldots & v_{P_r}(\alpha) \end{bmatrix}^T$ by $\mathrm{val}_\infty(\alpha)$. In order to define compact representation, we present the following definitions from [15].

Definition 2.39. [15, Definition 4.1] For an element $\alpha \in F$, and a vector $v = \begin{bmatrix} v_1 & \ldots & v_r \end{bmatrix}^T \in \mathbb{Q}^r$, we say that α is *close* to v if

$$\sum_{i=1}^{r} |v_{P_i}(\alpha) - v_i| \leq n + g.$$

Definition 2.40. [15, Definition 4.2] A *binary multiplicative representation (BMR)* of an element $\beta \in F$ is a sequence $(\beta_1, \ldots, \beta_l)$, where $\beta_i \in F$ such that

$$\beta = \prod_{i=1}^{l} \beta_i^{2^{l-i}}.$$

Now we present a definition of a compact representation $\mathbf{t}_\alpha$ of $\alpha \in F$ as in [15].

Definition 2.41. [15, Definition 4.3] A *compact representation* of $\alpha \in F$ is a pair $\mathbf{t}_\alpha = (\mu, (\beta_1, \beta_2, \cdots, \beta_l))$ where $\mu \in F$ such that $\alpha = \frac{\mu}{\beta}$ where β is a minimum of O_F represented by the BMR $(\beta_1, \beta_2, \cdots, \beta_l)$, and

1. $l \leq \log(\|\mathrm{val}_\infty(\alpha)\|_\infty + g)$,

2. The size of μ is polynomial in $\log q$, n, and $\mathbf{h}(\mathrm{Norm}(\alpha))$, and

3. The size of β_i is polynomial in $\log q$ and n.

Here size means the number of bits to represent the element.

That means, given a compact representation $\mathbf{t}_\alpha = (\mu, (\beta_1, \ldots, \beta_l))$ of $\alpha \in F$, we have

$$\alpha = \mu \prod_{i=1}^{l} \left(\frac{1}{\beta_i}\right)^{2^{l-i}}. \tag{2.30}$$

We provide examples of standard and compact representations of a fundamental unit of a global function field in Appendix A.

2.4.1 Computing compact representations

In this section, we summarize the algorithms for computing compact representations of Eisentrager and Hallgren in [15]. To compute compact representations with these algorithms, it is required for F to have at least one infinite place of degree 1. Throughout the description, we let P_{r+1} be a degree 1 infinite place of F.

We can compute a compact representation of $\alpha \in F$ when given the principal ideal $A = \alpha O_F$ generated by α in HNF representation and the vector $\mathrm{val}_\infty(\alpha)$. Computing a compact representation consists of two parts; one is to compute a minimum μ of the principal ideal A, and the other is to compute a BMR of a minimum β of O_F such that $\beta = \frac{\mu}{\alpha}$. We first present Algorithm 3, which is used in both parts, then present how to compute compact representations efficiently.

Algorithm 3 is a part of [15, Algorithm 3.4]. Algorithm 3 takes two inputs: an ideal I and an integer vector $v = \begin{bmatrix} v_1 & \ldots & v_r \end{bmatrix} \in \mathbb{Z}^r$. It returns a minimum $\gamma \in I$ that is close to v by computing Riemann-Roch spaces as follows. Let $D = \mathrm{div}(I) + \sum_{i=1}^{r} v_i P_i$. Then we compute the Riemann-Roch space $L(D + \ell P_{r+1})$ for $\ell \in [-\deg D, -\deg D + g]$ to find ℓ such that $L(D + \ell P_{r+1})$ is not trivial. With the smallest such ℓ, we pick γ from the basis of $L(D + \ell P_{r+1})$. The correctness of the algorithm is proved in [15, Proposition 3.5].

Algorithm 3 Reduce [15, Algorithm 3.4]

Input: An ideal I, a vector $v = (v_1, v_2, \cdots, v_r) \in \mathbb{Z}^r$
Output: a minimum γ of I that is close to v

1: $D \longleftarrow \mathrm{div}(I) + \sum_{i=1}^{r} v_i P_i$
2: **for** $\ell \in [-\deg D, -\deg D + g]$ **do**
3: **if** $L(D + \ell P_{r+1})$ is not trivial **then**
4: Pick γ from a basis of $L(D + \ell P_{r+1})$.
5: **return** γ
6: **end if**
7: **end for**

When computing a compact representation of $\alpha \in F$, we suppose that we have the ideal $A = \alpha O_F$ generated by α in HNF representation and the vector $\mathrm{val}_\infty(\alpha)$. As the first part of computing a compact representation of α, we find a minimum μ of A using Algorithm 3 with inputs A and $\vec{0} \in \mathbb{Z}^r$. Then Algorithm 3 returns $\mu \in A$ that is close to $\vec{0} \in \mathbb{Z}^r$ and $\frac{1}{\mu}A$ is a reduced ideal.

Let $\beta = \frac{\mu}{\alpha}$. Since $\frac{1}{\mu}A = \frac{\alpha}{\mu}O_F$ is an integral ideal, we know that β is in O_F. We compute a BMR of β by computing each β_i in the order using Algorithm 3 such that the interim BMR $(\beta_1, \ldots \beta_i)$ is close to $\frac{1}{2^{l-i}}\mathrm{val}_\infty(\beta)$, where $l = \lfloor \log_2 \|\mathrm{val}_\infty(\mu) - \mathrm{val}_\infty(\alpha)\|_\infty \rceil + 1$, where $\lfloor \cdot \rceil$ rounds up the real number inside to the nearest integer. To be specific, we start with computing $\beta_1 \in O_F$ that is close to the vector $\frac{1}{2^{l-1}}\mathrm{val}_\infty(\beta) = \frac{1}{2^{l-1}}\mathrm{val}_\infty\left(\frac{\mu}{\alpha}\right)$ by Algorithm 3 with inputs $1 \cdot O_F$ and the integer vector

$$t = \left[\left\lfloor \frac{v_{P_1}(\mu) - v_{P_1}(\alpha)}{2^{l-1}} \right\rceil \quad \cdots \quad \left\lfloor \frac{v_{P_r}(\mu) - v_{P_r}(\alpha)}{2^{l-1}} \right\rceil \right]^T \in \mathbb{Z}^r.$$

Then we compute each β_i for $2 \leq i \leq l$ as follows. Denote the element for which $(\beta_1, \ldots, \beta_{i-1})$ is the BMR by $\beta_{(i)}$. Then β_i is a minimum of $\frac{1}{\beta_{(i)}^2}O_F$ and is close to $\frac{1}{2^{l-i}}\mathrm{val}_\infty(\beta) - 2\mathrm{val}_\infty(\beta_{(i)})$. Eventually, β_l is a minimum of $\frac{1}{\beta_{(l)}^2}O_F = \frac{1}{\prod_{j=1}^{l-1}\beta_j^{2^{l-j}}}O_F$ that is close to $\mathrm{val}_\infty(\beta) - \mathrm{val}_\infty(\prod_{j=1}^{l-1}\beta_j^{2^{l-j}})$. [15, Corollary 4.13] shows that $(\beta_1, \ldots, \beta_l)$ is the BMR of β. Algorithm 4 shows the entire process of computing a compact representation of α when given the principal ideal αO_F in HNF representation and the vector $\mathrm{val}_\infty(\alpha)$. Algorithm 4 is simplified from [15, Algorithm

4.6, 4.8, 4.10].

Algorithm 4 CompRep

Input: A principal ideal $A = \alpha O_F$, a vector $\mathrm{val}_\infty(\alpha) = (v_{P_1}(\alpha), v_{P_2}(\alpha), \cdots, v_{P_r}(\alpha)) \in \mathbb{Z}^r$
Output: A compact representation of α

$\mu \longleftarrow$Reduce(A,$\vec{0}$)
$l \longleftarrow \lfloor \log_2 \|\mathrm{val}_\infty(\mu) - \mathrm{val}_\infty(\alpha)\|_\infty \rceil + 1$
$B \longleftarrow 1 \cdot O_F$, $\beta_0 \longleftarrow 1$, $v_\beta \longleftarrow \vec{0} \in \mathbb{Z}^r$
for $i \in \{1, \ldots, l\}$ **do**
  $t \longleftarrow \left[\lfloor (v_{P_1}(\mu) - v_{P_1}(\alpha))/2^{l-i} \rceil \quad \ldots \quad \lfloor (v_{P_r}(\mu) - v_{P_r}(\alpha))/2^{l-i} \rceil\right]^T$
  $B \longleftarrow 1/\beta_{i-1}^2 B^2$
  $\beta_i \longleftarrow$Reduce(B, $t - 2v_\beta$)
  $v_\beta \longleftarrow 2v_\beta + \mathrm{val}_\infty(\beta_i)$
end for
return $(\mu, (\beta_1, ...\beta_l))$;

2.4.2 Operations with compact representations

In this subsection, we provide algorithms to do multiplication, exponentiation, and compute the norm along with descriptions of how they work.

In order to describe arithmetic operations with compact representations, let $\mathbf{t}_1 = (\mu_1, (\beta_{1,1}, \ldots, \beta_{1,l_1}))$ and $\mathbf{t}_2 = (\mu_2, (\beta_{2,1}, \ldots, \beta_{2,l_2}))$ be compact representations of $\alpha_1 \in F$ and $\alpha_2 \in F$, respectively. This means, we have

$$\alpha_1 = \mu_1 \prod_{i=1}^{l_1} \left(\frac{1}{\beta_{1,i}}\right)^{2^{l_1-i}} \quad \text{and} \quad \alpha_2 = \mu_2 \prod_{i=1}^{l_2} \left(\frac{1}{\beta_{2,i}}\right)^{2^{l_2-i}}, \tag{2.31}$$

as in (2.30).

First, we consider computing a compact representation of $\alpha_1\alpha_2$ using $\mathbf{t}_1$ and $\mathbf{t}_2$. We have two cases; $l_1 = l_2$ and $l_1 \neq l_2$. When $l_1 = l_2$, from (2.31), we have

$$\alpha_1\alpha_2 = \mu_1\mu_2 \prod_{i=1}^{l_1} \left(\frac{1}{\beta_{1,i}\beta_{2,i}}\right)^{2^{l_1-i}}.$$

Thus, we simply multiply $\mathbf{t}_1$ and $\mathbf{t}_2$ componentwise to compute a new compact representation

of $\alpha_1\alpha_2$,

$$(\mu_1\mu_2,\ (\beta_{1,1}\beta_{2,1},\ \ldots,\ \beta_{1,l_1}\beta_{2,l_1}))\,.$$

On the other hand, when $l_2 \neq l_1$, without loss of generality, we assume that $l_1 < l_2$. Then we have

$$\alpha_1\alpha_2 = \mu_1\mu_2 \left(\prod_{i=1}^{l_2-l_1} \left(\frac{1}{\beta_{2,i}}\right)^{2^{l_2-i}}\right) \left(\prod_{i=l_2-l_1+1}^{l_2} \left(\frac{1}{\beta_{1,i-l_2+l_1}\beta_{2,i}}\right)^{2^{l_2-i}}\right).$$

Thus, the following is a compact representation of $\alpha_1\alpha_2$ in this case,

$$(\mu_1\mu_2,\ (\beta_{2,1},\ \ldots,\ \beta_{2,l_2-l_1},\ \beta_{1,1}\beta_{2,l_2-l_1+1},\ \ldots,\ \beta_{1,l_1}\beta_{2,l_2}))\,.$$

Algorithm 5 multiplies two compact representations as described above.

Algorithm 5 MultCR

Input: Compact representations $\mathbf{t}_1 = (\mu_1, (\beta_{1,1}, \ldots, \beta_{1,l_1}))$ of $\alpha_1 \in F$, and $\mathbf{t}_2 = (\mu_2, (\beta_{2,1}, \ldots, \beta_{2,l_1}))$ of $\alpha_2 \in F$

Output: a compact representation $\mathbf{t}$ of $\alpha_1 \cdot \alpha_2$.

1: **if** $l_1 > l_2$ **then**
2: **return** $(\mu_1\mu_2,\ (\beta_{1,1},\ \ldots,\ \beta_{1,l_1-l_2},\ \beta_{2,1}\beta_{1,l_1-l_2+1},\ \ldots,\ \beta_{2,l_2}\beta_{1,l_1}))$
3: **else if** $l_1 = l_2$ **then**
4: **return** $(\mu_1\mu_2,\ (\beta_{1,1}\beta_{2,1},\ \ldots,\ \beta_{1,l_1}\beta_{2,l_1}))$
5: **else**
6: **return** $(\mu_1\mu_2,\ (\beta_{2,1},\ \ldots,\ \beta_{2,l_2-l_1},\ \beta_{1,1}\beta_{2,l_2-l_1+1},\ \ldots,\ \beta_{1,l_1}\beta_{2,l_2}))$
7: **end if**

Now we consider computing a compact representation of α^m for some integer $m \in \mathbb{Z}$ when α is given in compact representation $\mathbf{t} = (\mu, (\beta_1, \ldots, \beta_l))$. With (2.31), we know that

$$\alpha^m = \left(\mu \prod_{i=1}^{l} \left(\frac{1}{\beta_i}\right)^{2^{l-i}}\right)^m = \mu^m \prod_{i=1}^{l} \left(\frac{1}{\beta_i^m}\right)^{2^{l-i}}.$$

Thus, $(\mu^m, (\beta_1^m, \ldots, \beta_l^m))$ is a compact representation of α^m. Algorithm 6 conducts the described compact representation exponentiation.

Computing the norm of α with its compact representation $\mathbf{t}$ can be done as follows. Due to the multiplicative form, we can compute the norm of α by computing the norms of each

Algorithm 6 PowCR

Input: Compact representation $\mathbf{t} = (\mu, (\beta_1, \ldots, \beta_l))$ of $\alpha \in F$, and an integer m
Output: a compact representation of α^m.

1: **return** $(\mu^m, (\beta_1^m, \ldots, \beta_l^m))$;

component of $\mathbf{t}$ as in the following equality,

$$\begin{aligned}\mathrm{Norm}_{F/k(x)}(\alpha) &= \mathrm{Norm}_{F/k(x)}\left(\mu \prod_{i=1}^{l}\left(\frac{1}{\beta_i}\right)^{2^{l-i}}\right) \\ &= \mathrm{Norm}_{F/k(x)}(\mu)\prod_{i=1}^{l}\left(\frac{1}{\mathrm{Norm}_{F/k(x)}(\beta_i)}\right)^{2^{l-i}}. \end{aligned} \tag{2.32}$$

Algorithm 7 computes the norm of an element in compact representation as described above.

Algorithm 7 NormCR

Input: Compact representation $\mathbf{t} = (\mu, (\beta_1, \ldots, \beta_l))$ of $\alpha \in F$
Output: $\mathrm{Norm}_{F/k(x)}(\alpha)$.

1: **return** $\mathrm{Norm}_{F/k(x)}(\mu)\prod_{i=1}^{l}\mathrm{Norm}_{F/k(x)}(\beta_i)^{-2^{l-i}}$

When computing values of α given in compact representation $\mathbf{t}$ at a place $P \in \mathbb{P}(F)$, with (2.31), we have

$$v_P(\alpha) = v_P(\mu) - \sum_{i=1}^{l} 2^{l-i} v_P(\beta_i).$$

Thus, $v_P(\alpha)$ can be computed by computing values of μ and β_i at P. Algorithm 8 computes the norm of an element in compact representation as described above.

Algorithm 8 ValCR

Input: Compact representations $\mathbf{t}_1$ of $\alpha_1 \in F$, a place P
Output: $v_P(\alpha_1)$

1: **return** $v_P(\mu_{\mathbf{t}_1}) - \sum_{j=1}^{l_1} 2^{l+1-j} v_P(\beta_{\mathbf{t}_1,j})$

Lastly, we give Algorithm 9 for testing whether or not two elements given in compact representation are associate. It is a straightforward modification of Algorithm 1 for compact representations. Algorithm 9 takes one more input, the set $S = \{P \in \mathbb{P}_0(F) \mid v_P(\alpha_1) \neq 0 \text{ or } v_P(\alpha_2) \neq 0\}$.

Algorithm 9 IsAssociateCR

Input: Compact representations $\mathbf{t}_1 = (\mu_1, (\beta_{1,1}, \ldots, \beta_{1,l_1}))$ of $\alpha_1 \in F$, and $\mathbf{t}_2 = (\mu_2, (\beta_{2,1}, \ldots, \beta_{2,l_1}))$ of $\alpha_2 \in F$, $S = \{P \in \mathbb{P}_0(F) \mid v_P(\alpha_1) \neq 0 \text{ or } v_P(\alpha_2) \neq 0\}$

Output: true if α_1 and α_2 are associate, false otherwise.

1: **if** $\mathrm{ValCR}(\mathbf{t}_1) - \mathrm{ValCR}(\mathbf{t}_2) = 0$ for $P \in S$ **then**
2: **return** true (associate)
3: **else**
4: **return** false (non-associate)
5: **end if**

2.4.3 Complexity analysis

In this section, we first discuss the cost of computing compact representations using Algorithm 4, and the sizes of compact representations. Later in this section, we analyze the cost of computing with compact representations. In particular, we provide the cost of Algorithms 5, 6, 7, 8, and 9. We keep the same assumptions as in Section 2.3. All computational costs are expressed in terms of the number of bit operations. Using the algorithms in [23], an operation in k takes $O((\log q)^{1+\varepsilon})$ bit operations.

We first analyze the cost of computing a compact representation. A brief analysis is given in [15] indicating the complexity is polynomial in the size of inputs, but this is given without exact factors. We conduct a thorough complexity analysis to use in later chapters to analyze our new algorithms accurately.

We recall the cost of computing a Riemann-Roch space, because it is an essential part of analyzing the cost of computing a compact representation. Let $D = \sum_{P \in \mathbb{P}(F)} a_P P$ be a divisor of F with bounded height, $\mathbf{h}(D) \leq h_D$ for some positive integer h_D. For brevity, we denote the cost of computing a k-basis of the Riemann-Roch space $L(D)$ of D by $\mathsf{RR}(h_D)$ from now on. Then by [5, Theorem 4.13], we have

$$\mathsf{RR}(h_D) = O\left(\left(n^5(h_D + n^2 C_f)^2 + n^{5+\varepsilon} C_f^{2+\varepsilon}\right) q^{\varepsilon}\right) \tag{2.33}$$

bit operations, with C_f defined as in (2.28). Note that $C_f = O(g)$ when $g \to \infty$ from the equation in [4, Corollary 3.5] and $\mathsf{RR}(mh_D) = \mathsf{RR}(h_D)$ for any constant $m \in \mathbb{R}$.

Now we investigate the cost of computing a compact representation, which is the cost of Algorithm 4. The cost of Algorithm 4 is greatly dependent on the cost of its subroutine, Algorithm 3. Thus, we start by analyzing the cost of Algorithm 3.

We consider Algorithm 3 with inputs A and v, where A is an ideal A of O_F and $v \in \mathbb{Z}^r$ is an integer vector. Algorithm 3 requires computing up to g Riemann-Roch spaces $L(D)$ of

$$D = \operatorname{div}(A) + \sum_{i=1}^{r} |v_i| P_i + \ell P_{r+1},$$

where $P_i \in \mathbb{P}_\infty(F)$ for $1 \leq i \leq r+1$, $\deg(P_{r+1}) = 1$, and ℓ is an integer in the interval

$$\ell \in [-\deg(\operatorname{div}(A)) - \sum_{i=1}^{r} v_i \deg P_i, -\deg(\operatorname{div}(A)) - \sum_{i=1}^{r} v_i \deg P_i + g]. \tag{2.34}$$

We have the general cost of computing a Riemann-Roch space in (2.33) with respect to $\mathbf{h}(D)$. In this case we have

$$\mathbf{h}(D) = \mathbf{h}(\operatorname{div}(A)) + \sum_{i=1}^{r} |v_i| \deg(P_i) + |\ell|.$$

To bound $\mathbf{h}(D)$ with the inputs of Algorithm 3, we use the following inequalities.

$$\begin{aligned} \sum_{i=1}^{r} |v_i| \deg P_i &\leq \|v\|_\infty \sum_{i=1}^{r} \deg P_i \\ &\leq \|v\|_\infty \sum_{i=1}^{r} e_{P_i} \deg P_i = \|v\|_\infty n. \end{aligned}$$

The first inequality is from $\|v\|_\infty = \max_i |v_i|$ and the second inequality is because the ramification indices $e_{P_i} \geq 0$. The last equality is by (2.3). We can also bound $|\ell|$ as follows.

$$\begin{aligned} |\ell| &\leq \max\{|-\deg(\operatorname{div}(A)) - \sum_{i=1}^{r} v_i \deg P_i|, |-\deg(\operatorname{div}(A)) - \sum_{i=1}^{r} v_i \deg P_i + g|\} \\ &\leq \mathbf{h}(\operatorname{div}(A)) + n\|v\|_\infty + g. \end{aligned}$$

Thus, the height of D is bounded as

$$\mathbf{h}(D) \leq 2(\mathbf{h}(\mathrm{div}(A)) + n \left\|v\right\|_\infty) + g.$$

Since Algorithm 3 computes Riemann-Roch spaces of such D g times, the total cost is

$$O\left(gq^{\varepsilon}\mathsf{RR}\left(2(\mathbf{h}(\mathrm{div}(A)) + n \left\|v\right\|_\infty) + g\right)\right) = O\left(gq^{\varepsilon}\mathsf{RR}\left(\mathbf{h}(\mathrm{div}(A)) + n \left\|v\right\|_\infty + g\right)\right) \tag{2.35}$$

bit operations.

Now we consider the cost of Algorithm 3 with a principal ideal input. When A is the principal ideal generated by an element $\alpha \in F$, we can bound $\mathbf{h}(\mathrm{div}(A)) = \mathbf{h}(\mathrm{div}(\alpha O_F))$ by the height of the norm of α. Let N_α denote $\mathrm{Norm}_{F/k(x)}(\alpha)$. Then

$$\begin{aligned}
\mathbf{h}(\mathrm{div}(A)) &= \sum_{P \in \mathbb{P}_0(F)} |v_P(\alpha)| \deg P \\
&\leq \sum_{P \in \mathbb{P}_0(F)} |v_P(N_\alpha)| \deg P \\
&\leq 2n\mathbf{h}(N_\alpha).
\end{aligned} \tag{2.36}$$

The inequalities are by Lemma 3.7 and Lemma 2.35 in order. Algorithm 3 with the principal ideal A takes

$$O(gq^{\varepsilon}\mathsf{RR}\left(4n\mathbf{h}(N_\alpha) + 2n \left\|v\right\|_\infty + g\right)) = O(gq^{\varepsilon}\mathsf{RR}\left(n\mathbf{h}(N_\alpha) + n \left\|v\right\|_\infty + g\right)) \tag{2.37}$$

bit operations.

With the analysis of the cost of Algorithm 3, we analyze the cost of computing a compact representation, which is the cost of Algorithm 4. Algorithm 4 takes two inputs, a principal ideal $A = \alpha O_F$ and an integer vector $\mathrm{val}_\infty(\alpha) \in \mathbb{Z}^r$. It computes a compact representation $(\mu, (\beta_1, \ldots, \beta_l))$, where $(\beta_1, \ldots, \beta_l)$ is a BMR of $\beta = \frac{\mu}{\alpha} \in F$. The algorithm consists of two

parts: computing a minimum μ of the input ideal and computing a BMR of β. We analyze the cost of Algorithm 4 by analyzing the cost of each part.

Firstly, the cost of computing μ is the cost of Algorithm 3 with inputs A and $\vec{0} \in \mathbb{Z}^r$. Using the cost of Algorithm 3 in (2.35), it takes

$$O(gq^{\varepsilon}\mathsf{RR}\left(\mathbf{h}(\mathrm{div}(A)) + g\right)) \tag{2.38}$$

bit operations.

Secondly, the cost of computing a BMR of β is the cost of the for-loop in Step 4-9 of Algorithm 4. The for-loop has 4 steps. The first and the last are integer vector arithmetic that are negligible compared to the others. The other two are computing the square of a reduced ideal and Algorithm 3. We first show that the cost of Steps 4-9 of Algorithm 4 is dominated by running Algorithm 3 $l := \lfloor \log_2 \left\| \mathrm{val}_\infty(\mu) - \mathrm{val}_\infty(\alpha) \right\|_\infty \rceil + 1$ times. In order to do that, we analyze the cost of computing HNF representations of the involved ideals and show it is less than Algorithm 3.

We start with investigating the cost of computing the HNF representation of I_0^2, where I_0 is a reduced ideal in the form of $\frac{1}{\prod_{j=1}^{i-1}\beta_j^{2^{l-j}}}O_F$.

Computing the HNF representation of I_0^2 can be obtained by computing the HNF representation of the product of two ideals. From [44, Theorem 5.2.2.], this costs $O\left(n^7 g^2 q^{\varepsilon}\right)$ bit operations, when n and g go to infinity, which is less than the cost of Algorithm 3 in (2.35). We bound $\deg(\mathrm{div}(I_0))$ using the bound in (2.6). Since I_0 is a reduced ideal, $\mathrm{div}(I_0)$ satisfies

$$\mathrm{div}(I_0) \geq 0, \text{ and } \deg(\mathrm{div}(I_0)) = \mathbf{h}(\mathrm{div}(I_0)) \leq g. \tag{2.39}$$

Now that we know the cost of Steps 4-9 of Algorithm 4 is dominated by running Algorithm 3 l times, we consider the cost of running Algorithm 3 with inputs I_0^2 and $t \in \mathbb{Z}^r$, where I_0 is a reduced ideal and t is an integer vector such that $\sum_{i=1}^{r} |(t)_i| \leq 3(n+g)$ [15, proof of

Lemma 4.7], thus $\|t\|_\infty \leq 3(n+g)$. The reduced ideal I_0 satisfies (2.39), which implies

$$\mathrm{div}(I_0^2) = 2\mathrm{div}(I_0) \geq 0, \text{ and } \deg(\mathrm{div}(I_0^2)) = \mathbf{h}(\mathrm{div}(I_0^2)) = 2\deg(\mathrm{div}(I_0)) \leq 2g.$$

Combining the above bounds and the cost of Algorithm 3 in (2.35), each call of Algorithm 3 in Step 4-9 of Algorithm 4 costs

$$O\left(gq^{\varepsilon}\mathsf{RR}\left(2g + n\|t\|_\infty + g\right)\right) = O\left(gq^{\varepsilon}\mathsf{RR}\left(n^2 + ng\right)\right)$$

bit operations. Since Step 4-9 of Algorithm 4 requires performing Algorithm 3 l times, Algorithm 4 takes

$$O\left(glq^{\varepsilon}\mathsf{RR}\left(n^2 + ng\right)\right) \tag{2.40}$$

bit operations.

Lastly, l can be bounded as follows. Since μ is an element close to the zero vector, we have $\|\mathrm{val}_\infty(\mu)\|_1 \leq n + g$ from Definition 2.39. That means $\|\mathrm{val}_\infty(\mu) - \mathrm{val}_\infty(\alpha)\|_\infty \leq \|\mathrm{val}_\infty(\alpha)\|_\infty + n + g$. Thus,

$$l = \lfloor \log_2(\|\mathrm{val}_\infty(\mu) - \mathrm{val}_\infty(\alpha)\|_\infty) \rceil + 1 \leq \lfloor \log_2(\|\mathrm{val}_\infty(\alpha)\|_\infty + n + g) \rceil + 1. \tag{2.41}$$

Finally, we have the cost of computing a compact representation of $\alpha \in F$ as in the following lemma.

Lemma 2.42. *Let A be a principal ideal of O_F generated by $\alpha \in F$ and $\mathrm{val}_\infty(\alpha) = \begin{bmatrix} v_{P_1}(\alpha) & \dots & v_{P_r}(\alpha) \end{bmatrix}^T \in \mathbb{Z}^r$. Then Algorithm 4 with inputs A and $\mathrm{val}_\infty(\alpha)$ takes*

$$O\left(gq^{\varepsilon}\left(\mathsf{RR}\left(n\mathbf{h}(N_\alpha) + g\right) + \log\left(\|\mathrm{val}_\infty(\alpha)\|_\infty + n + g\right)\mathsf{RR}\left(n^2 + ng\right)\right)\right)$$

bit operations, where $N_\alpha = \mathrm{Norm}_{F/k(x)}(\alpha)$.

Proof. Computing μ using Algorithm 3 with inputs $A = \alpha O_F$ and $\vec{0} \in \mathbb{Z}^r$ costs $O\left(g\mathsf{RR}\left(n\mathbf{h}(N_\alpha) + g\right)\right)$ bit operations, using the cost of Algorithm 3 in (2.37). The cost of computing a BMR of $\beta = \frac{\mu}{\alpha}$ is

$$O\left(gq^{\varepsilon}\log\left(\|\mathrm{val}_\infty(\alpha)\|_\infty + n + g\right)\mathsf{RR}\left(n^2 + ng\right)\right)$$

bit operations combining the cost in (2.40) and the bound of l in (2.41). □

We measure the size of a compact representation $\mathbf{t}_\alpha = (\mu, \beta_1, \ldots, \beta_{l_\alpha})$ of $\alpha \in F$ as the number of bits required to represent it. In order to do that, we first find bounds on $\mathbf{h}(\mu)$ and $\mathbf{h}(\beta_i)$, because the sizes of μ and β_i can be bounded by their height. We know that α and β_i belong to some Riemann-Roch spaces. The height of an element of F in a Riemann-Roch space is bounded as in the following lemma.

Lemma 2.43. *Let D be a divisor of F. If the Riemann-Roch space $L(D)$ is not trivial, for any element $\alpha \in L(D)$, the height of α is bounded*

$$\mathbf{h}(\alpha) = O(\mathbf{h}(D) + n^2 C_f).$$

Proof. Note that for any $\alpha \in L(D)$, α satisfies $v_P(\alpha) + v_P(D) \geq 0$ for any $P \in \mathbb{P}(F)$. That means

$$-\min_{P\in\mathbb{P}_\infty(F)}\{v_P(\alpha) + v_P(D)\} \leq 0.$$

Then by [6, Lemma 3.6], $\mathbf{h}(\alpha) = O(\mathbf{h}(D) + \delta_f)$, where δ_f is the degree of the discriminant of f. By [5, Lemma 4.10], we have $\delta_f \in O(n^2 C_f)$. □

With the above lemma, we can bound the heights of μ and β_i as follows.

Lemma 2.44. *Let $\mathbf{t}_\alpha = (\mu, \beta_1, \ldots, \beta_{l_\alpha})$ =CompRep(αO_F, $v_\infty(\alpha)$) be a compact representation of $\alpha \in F$, $N_\alpha = Norm_{F/k(x)}(\alpha) \in k(x)$, and C_f be as defined in (2.28). Then*

$l_\alpha = O(\log \|\mathrm{val}_\infty(\alpha)\|_\infty)$ *and the heights of* μ *and* β_i *are as follows,*

$$\mathbf{h}(\mu) = O(\mathbf{h}(\mathrm{div}(A)) + g + n^2 C_f) = O(n\mathbf{h}(N_\alpha) + g + n^2 C_f),$$
$$\mathbf{h}(\beta_i) = O(n^2 + ng + n^2 C_f) \quad \textit{for } 1 \le i \le l_\alpha.$$

Proof. From Algorithm 4, we know that μ is in the Riemann-Roch space $L(D)$, where

$$D = \mathrm{div}(\alpha O_F) + lP_{r+1}, \text{ where } l \in [-\deg(\mathrm{div}(\alpha O_F)), -\deg(\mathrm{div}(\alpha O_F)) + g].$$

Then we have

$$\begin{aligned}
\mathbf{h}(D) &= \mathbf{h}(\mathrm{div}(\alpha O_F)) + |l| \\
&\le 2\mathbf{h}(\mathrm{div}(\alpha O_F)) + g \\
&= 2 \sum_{P \in \mathbb{P}_0(F)} |v_P(\alpha)| \deg P + g \\
&\le 2 \sum_{P \in \mathbb{P}_0(F)} |v_P(N_\alpha)| \deg P + g \\
&\le 2n\mathbf{h}(N_\alpha) + g.
\end{aligned}$$

The last two inequalities are by Lemma 3.7 and Lemma 2.35 in order. By Lemma 2.43, $\mathbf{h}(\mu) = O(\mathbf{h}(D) + n^2 C_f)$. Thus, $\mathbf{h}(\mu) = O(n\mathbf{h}(N_\alpha) + g + n^2 C_f)$.

Now we analyze the height of β_i for $1 \le i \le l$. As discussed before, each β_i is an output of $\mathrm{Reduce}(I_0^2, t)$, where I_0 is a reduced ideal and t is a vector such that $\sum_{i=1}^{r} |(t)_i| \le 3(n+g)$ [15, proof of Lemma 4.7], thus $\|t\|_\infty \le 3(n+g)$. That means β_i is in the Riemann-Roch space $L(D_i)$ where

$$\mathbf{h}(D_i) = \mathbf{h}(I_0^2) + \mathbf{h}\left(\sum t_i P_i + \ell P_{r+1}\right) = O(g + n\|t\|_\infty) = O(n^2 + ng).$$

Thus, by Lemma 2.43, $\mathbf{h}(\beta_i) = O(n^2 + ng + n^2 C_f)$. □

From the height of μ and β_i, we bound the number of elements in k required to represent them, $2n\mathbf{h}(\mu)$ and $2n\mathbf{h}(\beta_i)$, respectively. Corollary 2.45 follows directly from Lemma 2.44 and the fact that $\log q$ bits are required to represent an element of k.

Corollary 2.45. *We suppose that* $\log q$ *bits are required to represent an element of* k*. Let* $\mathbf{t} = (\mu, \beta_1, \ldots, \beta_{l_\alpha})$ *be a compact representation of* α *computed by Algorithm 4. Then the numbers of bits required to represent* μ *and* β_i *are*

$$O\left((n^2\mathbf{h}(N_\alpha) + ng + n^3C_f)\log q\right) \quad \textit{and} \quad O\left((n^3 + n^2g + n^3C_f)\log q\right),$$

respectively.

Operations with compact representations

In this subsection, we discuss the cost of Algorithms 5, 6, and 7, which are algorithms for multiplying two compact representations, computing the m-th power of a compact representation, and computing the norm of a compact representation, respectively. We have discussed the general cost of performing multiplication, exponentiation of rational functions and elements of F, and the norm of an element of F in the beginning of this chapter. The following lemmas are direct applications of those.

Algorithm 5 takes two compact representations $\mathbf{t}_1$ and $\mathbf{t}_2$ of $\alpha_1 \in F$ and $\alpha_2 \in F$, and returns a compact representation $\mathbf{t}$ of $\alpha_1\alpha_2$. The cost of Algorithm 5 is as in the following lemma.

Lemma 2.46. *Let* $\mathbf{t}_1 = (\mu_1, \beta_{1,1}, \ldots, \beta_{1,l_{\mathbf{t}_1}})$ *and* $\mathbf{t}_2 = (\mu_2, \beta_{2,1}, \ldots, \beta_{2,l_{\mathbf{t}_2}})$ *be compact representations of* α_1 *and* $\alpha_2 \in F$ *respectively. Let* $d_{N_\alpha} = \max\{\mathbf{h}(Norm_{F/k(x)}(\alpha_1)), \mathbf{h}(Norm_{F/k(x)}(\alpha_2))\}$ *and* $l_{min} = \min\{l_{\mathbf{t}_1}, l_{\mathbf{t}_2}\}$, $l_{max} = \max\{l_{\mathbf{t}_1}, l_{\mathbf{t}_2}\}$*. Then Algorithm 5 computes a compact representation* $\mathbf{t} = (\mu, \beta_1, \ldots, \beta_{l_{max}})$ *of* $\alpha_1\alpha_2$ *in*

$$O(n^3q^\varepsilon\left((nd_{N_\alpha} + g + n^2C_f)^{1+\varepsilon} + l_{min}(n^2 + ng + n^2C_f)^{1+\varepsilon}\right))$$

bit operations, and the height of μ and β_i are bounded by

$$\mathbf{h}(\mu) = O(nd_{N_\alpha} + g + n^2 C_f)$$
$$\mathbf{h}(\beta_i) = O(n^2 + ng + n^2 C_f) \text{ for } 1 \leq i \leq l_{max}.$$

Proof. Algorithm 5 performs one multiplications of μ_1 and μ_2, and l multiplications of $\beta_{1,i}$ and $\beta_{2,i}$. We have $\mathbf{h}(\beta_{1,i}), \mathbf{h}(\beta_{2,i}) = O(n^2 + ng + n^2 C_f)$, $\mathbf{h}(\mu_1) = O(n\mathbf{h}(\mathrm{Norm}_{F/k(x)}(\alpha_1)) + g + n^2 C_f)$, and $\mathbf{h}(\mu_2) = O(n, \mathbf{h}(\mathrm{Norm}_{F/k(x)}(\alpha_2)) + g + n^2 C_f)$ from Lemma 2.44. Note that the last two are $O(nd_{N_\alpha} + g + n^2 C_f)$, which is clearly bigger than $2\left(\left\lceil \frac{2g-1}{n} \right\rceil + 1\right)$. By Lemma 2.32, Computing $\mu_1 \mu_2$ takes

$$O(n^3 \mathcal{M}(nd_{N_\alpha} + g + n^2 C_f)) = O(n^3 q^\varepsilon (nd_{N_\alpha} + g + n^2 C_f)^{1+\varepsilon})$$

bit operations, and computing $\beta_{1,i}\beta_{2,i}$ for each i takes

$$O(n^3 \mathcal{M}(n^2 + ng + n^2 C_f)) = O(n^3 q^\varepsilon (n^2 + ng + n^2 C_f)^{1+\varepsilon})$$

bit operations. Since we compute $\beta_{1,i}\beta_{2,i}$ only for $1 \leq i \leq l_{min}$, the result follows. □

Algorithm 6 takes a compact representation $\mathbf{t}$ of $\alpha \in F$ and $m \in \mathbb{Z}$, and returns a compact representation of α^m. The cost of Algorithm 6 is as in the following lemma.

Lemma 2.47. *Let $\mathbf{t} = (\mu, \beta_1, \ldots, \beta_{l_{\mathbf{t}_1}})$ be a compact representations of $\alpha \in F$ with the height of the norm $d_{N_\alpha} = \mathbf{h}(Norm_{F/k(x)}(\alpha))$, and $m \in \mathbb{Z}$ be a positive integer. Then Algorithm 6 computes a compact representation of α^m in*

$$O\left(n^3 m^{1+\varepsilon} q^\varepsilon \left((nd_{N_\alpha} + g + n^2 C_f)^{1+\varepsilon} + l_{\mathbf{t}_1}(n^2 + ng + n^2 C_f)^{1+\varepsilon}\right) \log m\right)$$

bit operations, and the heights of μ and β_i are bounded

$$\mathbf{h}(\mu) = O(m(nd_{N_\mu} + g + n^2 C_f))$$
$$\mathbf{h}(\beta_i) = O(m(n^2 + ng + n^2 C_f)) \quad \textit{for} \quad 1 \le i \le l_{\mathbf{t}_1}.$$

Proof. The heights of μ and β_i are $O(nd_{N_\alpha} + g + n^2 Cf)$ and $O(n^2 + ng + n^2 C_f)$ as in the proof of Lemma 2.46. Algorithm 6 computes the m-th powers of μ_1 and $\beta_{1,i}$. It follows from Lemma 2.34, computing μ^m and $\beta_{1,i}^m$ takes $8n^3\mathcal{M}(m(nd_{N_\alpha} + g + n^2 C_f)) \log m$ and $8n^3\mathcal{M}(m(n^2 + ng + n^2 C_f)) \log m$ bit operations respectively. Adding and simplifying the costs, we obtain the result. □

Algorithm 7 takes a compact representation $\mathbf{t}$ of $\alpha \in F$, and returns its norm. To analyze the cost of computing the norm of a compact representation, we recall Lemma 2.36, the cost of computing the norm of an element. Using the result, we show the costs of computing the norm of an element in a compact representation in the following lemma.

Lemma 2.48. *Let $\mathbf{t} = (\mu, \beta_1, \ldots, \beta_{l_\mathbf{t}})$ be a compact representation of $\alpha \in F$, and $d_\mathbf{t} = \max\{\mathbf{h}(\mu), \max_{1 \le i \le l_\mathbf{t}}\{\mathbf{h}(\beta_{1,i})\}\}$. Algorithm 7 computes the norm of α in*

$$O(\left(n^3 l_\mathbf{t} d_\mathbf{t}^{1+\varepsilon} + l_\mathbf{t}(l_\mathbf{t} 2^{l_\mathbf{t}} d_\mathbf{t})^{1+\varepsilon}\right) q^\varepsilon)$$

bit operations.

Proof. Algorithm 7 computes the norm of α_1 by computing the norm of each element of $\mathbf{t}$ and multiplying the norms. Let $d_\beta = \max_{1 \le i \le l_\mathbf{t}}\{\mathbf{h}(\beta_i)\}$. Computing the norms of μ_1 and β_i takes $O(n^3 q^\varepsilon \mathbf{h}(\mu)^{1+\varepsilon})$ and $O(n^3 q^\varepsilon d_{\beta_{1,i}}^{1+\varepsilon})$ bit operations, respectively, by Lemma 2.36. Then we compute the norm of α as in (2.32). Computing each $\mathrm{Norm}_{F/k(x)}(\beta_i)^{2^{l_\mathbf{t}-i}}$ takes $4\mathcal{M}\left((2^{l_\mathbf{t}-i} - 1)d_\mathbf{t}\right)(l_\mathbf{t} - i)$ by Lemma 2.33. Thus, it multiplying the norms of μ and β_i to

get the norm takes

$$2l_{\mathbf{t}}\mathcal{M}(l_{\mathbf{t}}2^{l_{\mathbf{t}}}d_{\mathbf{t}}) + \sum_{i=1}^{l_{\mathbf{t}}} 4\mathcal{M}\left((2^{l_{\mathbf{t}}-i}-1)d_{\mathbf{t}}\right)(l_{\mathbf{t}}-i) = O(l_{\mathbf{t}}q^{\varepsilon}(l_{\mathbf{t}}2^{l_{\mathbf{t}}}d_{\mathbf{t}})^{1+\varepsilon})$$

bit operations by Lemma 2.31. In total, the cost of Algorithm 7 is $O(\left(l_{\mathbf{t}}n^3d_{\mathbf{t}}^{1+\varepsilon} + l_{\mathbf{t}}(l_{\mathbf{t}}2^{l_{\mathbf{t}}}d_{\mathbf{t}})^{1+\varepsilon}\right)q^{\varepsilon})$ bit operations. □

The cost of Algorithm 8 and Algorithm 9 follow directly from the cost of computing values of elements of F given in (2.26).

Algorithm 8 takes a compact representation $\mathbf{t} = (\mu, \beta_1, \ldots, \beta_{l_{\mathbf{t}}})$ of $\alpha \in F$ and a place P, and returns the value of α at P. Algorithm 8 computes the value of each component of $\mathbf{t}$ at P. Thus, Algorithm 8 computes the value of α at P in

$$O(l_{\mathbf{t}}q^{\varepsilon}(d_{N_\alpha}^{\omega} + d_{\alpha}^{1+\varepsilon}))$$

bit operations, where $d_{N_\alpha} = \mathbf{h}(Norm_{F/k(x)}(\alpha))$ and ω is the matrix multiplication exponent.

Algorithm 9 takes two compact representations $\mathbf{t}_1$ of $\alpha_1 \in F$, $\mathbf{t}_2$ of $\alpha_2 \in F$, and a finite set S of places, and returns whether or not they are associate. Algorithm 9 computes the values of α_1 and α_2 at the places in S and compares them. Let $\mathbf{t}_1 = (\mu_1, \beta_{1,1}, \ldots, \beta_{1,l_{\mathbf{t}_1}})$ and $\mathbf{t}_2 = (\mu_2, \beta_{2,1}, \ldots, \beta_{2,l_{\mathbf{t}_2}})$ be compact representations of α_1 and $\alpha_2 \in F$ respectively. Let $d_{N_\alpha} = \max\{\mathbf{h}(\mathrm{Norm}_{F/k(x)}(\alpha_1)), \mathbf{h}(\mathrm{Norm}_{F/k(x)}(\alpha_2))\}$ and $l_{max} = \max\{l_{\mathbf{t}_1}, l_{\mathbf{t}_2}\}$. Then Algorithm 9 determines whether or not α_1 and α_2 are associate in

$$O(l_{max}q^{\varepsilon}(d_{N_\alpha}^{\omega} + d_{\alpha}^{1+\varepsilon})) \tag{2.42}$$

bit operations.

Chapter 3

Solving norm equations: Gaál-Pohst approach

In this chapter, we describe an approach to solving norm equations in the maximal order O_F that was introduced by Gaál and Pohst [21]. The idea is that we compute bounds related to solutions of a norm equation (2.7) and test all elements within the bounds to find a set of non-associate solutions of (2.7). We call the set of elements within the bounds the search space of (2.7).

While Gaál and Pohst developed the idea in [21], the height bounds on the solutions were not explicitly given. We provide Algorithm 10 with details of the approach including explicit bounds in Lemma 3.2 and Lemma 2.27 in Section 3.1.1. Later in Section 3.2.1, we introduce a new algorithm that shares the idea of Gaál and Pohst's approach, but also uses compact representations. Using compact representations, Algorithm 11 has a smaller number of elements in the search space compared to Algorithm 10, and the sizes of the solutions are smaller because they are in compact representation. Finally, we analyze the complexity of Algorithms 10 and 11 in Section 3.1.2 and Section 3.2.2, respectively, showing that Algorithm 11 has better asymptotic complexity than Algorithm 10.

We use the same notation as in the previous chapter. Given $c \in k[x] \setminus k$, we define a

norm equation as in Definition 2.20. Our goal is to find all non-associate solutions of (2.7) in O_F, which are those elements $\alpha \in O_F$ that satisfy the norm equation up to a constant, i.e. satisfy $\mathrm{Norm}_{F/k(x)}(\alpha) = c\zeta$ for any constant $\zeta \in k^\times$.

3.1 Gaál-Pohst approach

This section includes the approach of solving norm equations in the maximal order O_F that was introduced in [21] along with explicit bounds in Section 3.1.1 and the complexity of the algorithm in Section 3.1.2.

3.1.1 Algorithm description

In this section, we first summarize the approach of solving norm equations in the maximal order O_F that is introduced in [21], which is an exhaustive search. Since the search space is not explicitly given in [21], we specify a search space for the algorithm. A set of non-associate solutions of (2.7) is computed by checking the norms of all elements of O_F in the search space. Then we provide the entire process in algorithmic form, Algorithm 10, which is suitable for implementation and complexity analysis. At the end of this section, a small example is provided for illustrative purposes.

Given $c \in k[x] \setminus k$, first, we suppose that there is a solution α of the norm equation (2.7), and represent it in the standard representation with respect to a reduced basis $\mathcal{B} = \{\omega_1, \ldots, \omega_n\}$ of O_F,

$$\alpha = \sum_{i=1}^{n} \lambda_i \omega_i, \tag{3.1}$$

with some $\lambda_i \in k[x]$ for $1 \leq i \leq n$. Then we compute an upper bound on the degree of each λ_i using the maximum norms of ω_i. Once upper bounds on the degrees of λ_i are determined for all $1 \leq i \leq n$, we enumerate all polynomials $\lambda_i \in k[x]$ within the degree bounds to construct α in the above form. Then we compute the norm of each α to check whether or not α is a solution of the norm equation. Since we only collect non-associate solutions, we

run Algorithm 1 whenever we find a new solution to confirm that the new solution is not associate to any of the solutions already found. If there is no solution of the form (3.1) within the computed degree bounds of λ_i, it is because our initial assumption is false, thus the norm equation has no solution.

Suppose that there exists a solution α of the norm equation (2.7) in O_F, represented as in (3.1). We now derive bounds on the coefficients λ_i. For brevity, we denote the maximum additive norm $\| \ \|_{\infty,+}$ defined in (2.12) by $\| \ \|_\infty$ throughout. Then since $\mathcal{B}$ is a reduced basis of O_F, we have

$$\|\alpha\|_\infty = \max_{1\le i\le n}\{\deg(\lambda_i) + \|\omega_i\|_\infty\}, \tag{3.2}$$

as in equation (2.14). In addition, Lemma 2.27 shows that the degree of each λ_i is bounded,

$$\deg(\lambda_i) \le \|\alpha\|_\infty - \|\omega_i\|_\infty.$$

We can precompute $\|\omega_i\|_\infty$ for $1 \le i \le n$, but we do not know $\|\alpha\|_\infty$ as α is a solution of the norm equation that we try to solve. Therefore, we let α be minimal among its associate elements with respect to the maximum norm. We compute an upper bound on $\|\alpha\|_\infty$.

First, we note that with the maximum additive norm defined in (2.12), we have $\|\alpha\|_\infty = \max_{P_\infty|p_\infty}\left\{-\frac{v_{P_\infty}(\alpha)}{e_{P_\infty|p_\infty}}\right\}$. So finding a bound for $v_{P_\infty}(\alpha)$ is sufficient for obtaining a bound on $\|\alpha\|_\infty$. To find a bound on $v_{P_\infty}(\alpha)$, let ε_j for $1 \le j \le r$ be a system of fundamental units of F. Then we have the following lemma for $v_{P_\infty}(\alpha)$.

Lemma 3.1. *If α is a solution of (2.20), then the following equation from [21, p.244] holds for some $x_j \in \mathbb{R}$ for $1 \le j \le r$. For any $P_\infty \in \mathbb{P}_\infty(F)$ above $p_\infty \in \mathbb{P}_\infty(k(x))$,*

$$v_{P_\infty}(\alpha) = \sum_{j=1}^{r} x_j v_{P_\infty}(\varepsilon_j) + \frac{1}{n} v_{P_\infty}(c) = \sum_{j=1}^{r} x_j v_{P_\infty}(\varepsilon_j) - \frac{e_{P_\infty|p_\infty}}{n}\deg(c), \tag{3.3}$$

where $\{\varepsilon_j | 1 \le j \le r\}$ is a set of fundamental units of O_F.

Proof. Similar to the lattice in (2.18), with the map

$$\phi : F^{\times} \longrightarrow \mathbb{Z}^{|r+1|}, \qquad \beta \mapsto \left(v_{P_{\infty,1}}(\beta), \ldots, v_{P_{\infty,r+1}}(\beta)\right),$$

$(\phi(O_F^{\times}), \| \ \|)$ is also a lattice in $\mathbb{Z}^{r+1}$ with the Euclidean norm $\| \ \|$, and the set $\{\phi(\varepsilon_1), \ldots, \phi(\varepsilon_r)\}$ is a basis of the lattice, and is also $\mathbb{R}$-linearly independent in $\mathbb{R}^{r+1}$ [33, Lemma in p.360].

We first show that the basis of the lattice and the vector $v_1 = \begin{bmatrix} = 1 & 1 & \ldots & 1 \end{bmatrix}^T \in \mathbb{Z}^{r+1}$ are $\mathbb{R}$-linearly independent using the proof by contradiction, thus $\{\phi(\varepsilon_1), \ldots, \phi(\varepsilon_r), v_1\}$ is a $\mathbb{Z}$-basis of $\mathbb{R}^{r+1}$. Suppose that the set is linearly dependent. Then v_1 can be written as a linear combination of $\phi(\varepsilon_i)$,

$$v_1 = \sum_{i=1}^{r} a_i \phi(\varepsilon_i)$$

for some $a_i \in \mathbb{R}$, which implies $\sum_{i=1}^{r} a_i v_{P_{\infty,j}}(\varepsilon_i) = 1$ for all $1 \leq j \leq r+1$. Using this equality, we have $\deg P_{\infty,j} = \sum_{i=1}^{r} a_i v_{P_{\infty,j}}(\varepsilon_i) \deg P_{\infty,j}$ for any $1 \leq j \leq r+1$, thus

$$\sum_{j=1}^{r+1} \deg P_{\infty,i} = \sum_{j=1}^{r+1} \sum_{i=1}^{r} a_i v_{P_{\infty,j}}(\varepsilon_i) \deg P_{\infty,j},$$

which is a contradiction because the left hand side is positive but the right hand side is 0 because $\deg \operatorname{div}(\varepsilon_i) = \sum_{j=1}^{r+1} v_{P_{\infty,j}}(\epsilon_i) \deg P_{\infty,j} = 0$ for all $1 \leq i \leq r$. Therefore, $\{\phi(\varepsilon_1), \ldots, \phi(\varepsilon_r), v_1\}$ is a $\mathbb{Z}$-basis of $\mathbb{R}^{r+1}$.

Now, suppose that α is a solution of (2.7), we consider a vector $(v_{P_{\infty,1}}(\alpha), \ldots, P_{\infty,r+1}(\alpha)) \in \mathbb{R}^{r+1}$. Then the vector must be a linear combination of $\{\phi(\varepsilon_1), \ldots, \phi(\varepsilon_r), v_1\}$, thus we have

$$v_{P_{\infty,j}}(\alpha) = \left(\sum_{i=1}^{r} x_i v_{P_{\infty,i}}(\varepsilon_i) \right) + x_{r+1},$$

for some $x_i \in \mathbb{R}$. From this, we have the following equalities for $v_{P_{\infty,j}}(c)$.

$$\begin{aligned} v_{P_{\infty,j}}(c) &= v_{P_{\infty,j}}(\mathrm{Norm}_{F/k(x)}(\alpha)) \\ &= \sum_{k=1}^{n} v_{P_{\infty,j}}(\sigma_k(\alpha)) \\ &= \left(\sum_{k=1}^{n} \sum_{i=1}^{r} x_i v_{P_{\infty,i}}(\sigma_k(\varepsilon_i)) \right) + n x_{r+1} \\ &= \left(\sum_{i=1}^{r} x_i v_{P_{\infty,i}}(\mathrm{Norm}_{F/k(x)}(\varepsilon_i)) \right) + n x_{r+1} \\ &= n x_{r+1}, \end{aligned}$$

where σ_i are embeddings of F. The last equality is because $\mathrm{Norm}_{F/k(x)}(\varepsilon_i) \in \mathbb{F}_q$. Thus, we have

$$x_{r+1} = \frac{v_{P_{\infty,j}}(c)}{n} = \frac{e_{P_{\infty,j}|p_\infty}}{n} \deg c.$$

□

Since we can compute $v_{P_\infty}(\varepsilon_j)$ and $\deg(c)$, we want to bound x_j. This is where we consider a smallest associate solution.

Let α_0 be a solution of the norm equation, and we want to find a smallest solution α that is associate to α_0. Since

$$v_{P_\infty}(\alpha_0) = \sum_{j=1}^{r} x_{\alpha_0,j} v_{P_\infty}(\varepsilon_j) - \frac{e_{P_\infty|p_\infty}}{n} \deg(c)$$

for some $x_{\alpha_0,j} \in \mathbb{R}$ as in (3.3), we construct the unit $u = \prod_{j=1}^{r} \varepsilon_j^{-\lfloor x_{\alpha_0,j} \rceil}$ where $\lfloor x_{\alpha_0,j} \rceil$ denotes the nearest integer to the real number $x_{\alpha_0,j}$ (rounding up for half integers). By multiplying

u with α_0, we get an associate solution α whose coefficients of all $v_{P_\infty}(\varepsilon_j)$ are bounded

$$\begin{aligned} v_{P_\infty}(\alpha) &= v_{P_\infty}(\alpha_0 \cdot u) = v_{P_\infty}(\alpha_0) + v_{P_\infty}(u) \\ &= \sum_{j=1}^{r} (x_{\alpha_0,j} - \lfloor x_{\alpha_0,j} \rceil) v_{P_\infty}(\varepsilon_j) - \frac{e_{P_\infty|p_\infty}}{n} \deg(c). \end{aligned} \tag{3.4}$$

Clearly, $-1/2 \leq x_{\alpha_0,j} - \lfloor x_{\alpha_0,j} \rceil \leq 1/2$ for $1 \leq j \leq r$. Using this bound on $x_{\alpha_0,j} - \lfloor x_{\alpha_0,j} \rceil$, we find bounds of $\|\alpha\|_\infty$, and eventually of $\deg(\lambda_i)$ in the following lemmas.

Lemma 3.2. *For α as given in* (3.4)*,*

$$\|\alpha\|_\infty \leq \Theta$$

where

$$\Theta = \max_{P_\infty|p_\infty} \left\{ \frac{\sum_{j=1}^{r} \frac{1}{2} |v_{P_\infty}(\varepsilon_j)|}{e_{P_\infty|p_\infty}} \right\} + \frac{1}{n} \deg(c)$$

Proof. From equation (3.4), we get the inequality

$$v_{P_\infty}(\alpha) \geq \sum_{j=1}^{r} \left(-\frac{1}{2} |v_{P_\infty}(\varepsilon_j)| \right) - \frac{e_{P_\infty|p_\infty}}{n} \deg(c).$$

By dividing both sides by $-e_{P_\infty|p_\infty}$, we obtain

$$-\frac{v_{P_\infty}(\alpha)}{e_{P_\infty|p_\infty}} \leq \frac{\sum_{j=1}^{r} \frac{1}{2} |v_{P_\infty}(\varepsilon_j)| + \frac{e_{P_\infty|p_\infty}}{n} \deg(c)}{e_{P_\infty|p_\infty}}.$$

Note that the right hand side of the inequality above is Θ. □

Lemma 3.3. *For any integer i such that $1 \leq i \leq n$, each λ_i in the equation (3.1) has bounded degree in x as below,*

$$\deg(\lambda_i) \leq \Theta - \Psi_i, \quad (1 \leq i \leq n)$$

where Θ is as defined in Lemma 3.2, and

$$\Psi_i = \|\omega_i\|_\infty = \max_{P_\infty | p_\infty} \left\{ -\frac{v_{P_\infty}(\omega_i)}{e_{P_\infty | p_\infty}} \right\}$$

Proof. We obtain the following inequality,

$$\begin{aligned} \deg(\lambda_i) &\leq \|\alpha\|_\infty - \|\omega_i\| \\ &\leq \Theta - \Psi_i, \end{aligned}$$

by combining Lemmas 2.27 and 3.2. □

Now we have explicitly computable bounds on the degrees of λ_i. In addition, we note that we only need the values of a system of fundamental units at the infinite places of F, not the actual units, which require a large amount of storage space. We obtain these values using Algorithm 2. After computing the bounds of Lemma 3.3, we compute the norm of each α of the form of (3.1) with λ_i of degree within the bounds. Then we remove associate solutions using Algorithm 1. Algorithm 10 describes the entire process.

Algorithm 10 NormEquation-GaalPohst

Input: $c \in k[x] \setminus \{0\}$, a reduced basis $\mathcal{B} = \{\omega_i | 1 \leq i \leq n\}$ of O_F, a unit value matrix $M_{\mathbb{P}_\infty(F)}$

Output: A set $\mathcal{R}$ of all non-associate solutions of the equation (2.7) in standard representation

1: $\mathcal{R} \longleftarrow \emptyset$
2: $v_l \longleftarrow \max_{P_j | p_\infty} \{\sum_{i=1}^{r} \frac{1}{2e_{P_j | p_\infty}} |(M_{\mathbb{P}_\infty(F)})_{i,j}|\}$
3: $v_c \longleftarrow \frac{1}{n} \deg(c)$
4: **for** $\alpha_{temp} = \sum_{i=1}^{n} \lambda_i \omega_i$ where $\deg \lambda_i \leq v_l + v_c - \|\omega_i\|_\infty$ for $1 \leq i \leq n$ **do**
5: **if** $\mathrm{Norm}_{F/k(x)}(\alpha_{temp})$/c in $k^\times$ **then**
6: **if** not (IsAssociate(α_{temp}, α)) for all $\alpha \in \mathcal{R}$ **then**
7: $\mathcal{R} \longleftarrow \mathcal{R} \cup \{\alpha_{temp}\}$
8: **end if**
9: **end if**
10: **end for**
11: **return** $\mathcal{R}$

The following is small examples of solving a norm equation using Algorithm 10.

Example 3.1.1. Let $F/\mathbb{F}_5(x)$ be a finite extension of degree $n = 3$ by

$$f(t) = t^3 + (x+2)t^2 + (x+2)t + 4x^2 + 3x + 2.$$

Then the genus of F is 1, and the ideal class number of F is 1. There are two infinite places $\{P_{\infty,1}, P_{\infty,2}\}$ of F, with ramification indices $e_{P_{\infty,1}|p_\infty} = 1$, $e_{P_{\infty,2}|p_\infty} = 2$. So F has unit rank 1.

Let $c = x + 2$. We want to find all non-associate elements in O_F that satisfy

$$\mathrm{Norm}_{F/\mathbb{F}_5(x)}(\alpha) = \zeta c = \zeta(x+2) \tag{3.5}$$

for some $\zeta \in \mathbb{F}_5^\times$.

Let $\omega_1 = 1$, $\omega_2 = y$, and $\omega_3 = y^2 + xy$. Then $\mathcal{B} = \{\omega_1, \omega_2, \omega_3\}$ is a reduced basis of O_F, where $\|\omega_1\|_\infty = 0$, $\|\omega_2\|_\infty = 1$, $\|\omega_3\|_\infty = \frac{3}{2}$. An element of O_F can be represented in the standard representation with respect to the reduced basis $\mathcal{B}$ as $\sum_{i=1}^3 \lambda_i \omega_i$ with polynomials $\lambda_i \in \mathbb{F}_5[x]$. Now, we suppose we have a precomputed unit value matrix using Algorithm 2,

$$M_{\mathbb{P}_\infty(F)} = \mathrm{SValMat}(\mathbb{P}_\infty(F)) = \begin{bmatrix} 4 & -4 \end{bmatrix}.$$

We denote a unit of O_F that corresponds to the row of $M_{P_\infty(F)}$ by ε. Then ε satisfies $v_{P_{\infty,1}}(\varepsilon) = 4$ and $v_{P_{\infty,2}}(\varepsilon) = -4$. Using $M_{P_\infty(F)}$ and Lemma 3.3, we compute Θ and Ψ_i to compute degree bounds for λ_i as below.

$$\begin{aligned}
\Theta &= \max_{P_\infty | p_\infty} \left\{ \frac{\frac{1}{2}|v_{P_\infty}(\varepsilon)| + \frac{e_{P_\infty|p_\infty}}{n} \deg(c)}{e_{P_\infty}|p_\infty} \right\} \\
&= \max\left\{ \frac{4}{2} + \frac{1}{3}, \quad \frac{|-4|}{2} + \frac{1}{3} \right\} \\
&= 2 + \frac{1}{3},
\end{aligned}$$

$$\Psi_1 = \|\omega_1\|_\infty = 0, \qquad \Psi_2 = \|\omega_2\|_\infty = 1, \qquad \text{and} \qquad \Psi_3 = \|\omega_3\|_\infty = \frac{3}{2}$$

Using Lemma 3.3, we can compute degree bounds for each λ_i as follows:

$$\deg(\lambda_1) \leq 2 + \frac{1}{3} - 0 = 2 + \frac{1}{3} \tag{3.6}$$

$$\deg(\lambda_2) \leq 2 + \frac{1}{3} - 1 = 1 + \frac{1}{3} \tag{3.7}$$

$$\deg(\lambda_3) \leq 2 + \frac{1}{3} - \frac{3}{2} = \frac{12 + 2 - 9}{6} = \frac{5}{6} \tag{3.8}$$

When searching for solutions, we compute the norms of $5^{3+2+1} = 5^6$ elements of the form of $\sum_{i=1}^{3} \lambda_i \omega_i$ with polynomials λ_1, λ_2, and λ_3 of degrees bounded as in (3.6), (3.7), and (3.8). By testing all elements, we find one solution of (3.5) up to associates, namely $\alpha = y + x$. The entire process took 1.140 CPU seconds on an Intel Xeon CPU E7-8891 v4 with 80 64-bit cores at 2.80GHz.

Example 3.1.2. Let $F/\mathbb{F}_5(x)$ be a finite extension of degree $n = 3$ by

$$f(t) = t^3 + (4x^3 + 3x^2 + 1)t^2 + (3x^3 + 4x^2 + 4x + 2)t + 2x^3 + x.$$

Then the genus of F is 4, and the ideal class number of F is 2. There are two infinite places $\{P_{\infty,1}, P_{\infty,2}\}$ of F, with ramification indices $e_{P_{\infty,1}|p_\infty} = e_{P_{\infty,2}|p_\infty} = 1$. So F has unit rank 1.

Let $c = x + 4$. We want to find all non-associate elements in O_F that satisfy

$$\mathrm{Norm}_{F/\mathbb{F}_5(x)}(\alpha) = \zeta c = \zeta(x + 4) \tag{3.9}$$

for some $\zeta \in \mathbb{F}_5^\times$.

Let $\omega_1 = 1$, $\omega_2 = y^2 + (4x^3 + 3x^2)y$, and $\omega_3 = y$. Then $\mathcal{B} = \{\omega_1, \omega_2, \omega_3\}$ is a reduced basis of O_F, where $\|\omega_1\|_\infty = 0$, $\|\omega_2\|_\infty = 3$, $\|\omega_3\|_\infty = 3$. An element of O_F can be represented in the standard representation with respect to the reduced basis $\mathcal{B}$ as $\sum_{i=1}^{3} \lambda_i \omega_i$ with polynomials $\lambda_i \in \mathbb{F}_5[x]$. Now, we suppose that we have a precomputed unit value matrix using Algorithm

2,

$$M_{\mathbb{P}_\infty(F)} = \mathrm{SValMat}(\mathbb{P}_\infty(F)) = \begin{bmatrix} 694 & -347 \end{bmatrix}.$$

We denote a unit of O_F that corresponds to the row of $M_{P_\infty(F)}$ by ε. Then ε satisfies $v_{P_{\infty,1}}(\varepsilon) = 694$ and $v_{P_{\infty,2}}(\varepsilon) = -347$. Using $M_{P_\infty(F)}$ and Lemma 3.3, we compute Θ and Ψ_i to compute degree bounds for λ_i as below.

$$\begin{aligned}
\Theta &= \max_{P_\infty | p_\infty} \left\{ \frac{\frac{1}{2}|v_{P_\infty}(\varepsilon)| + \frac{e_{P_\infty}|p_\infty}{n} \deg(c)}{e_{P_\infty}|p_\infty} \right\} \\
&= \max\left\{ \frac{1}{2}(694) + \frac{1}{3}, \quad \frac{1}{2}(|-347|) + \frac{1}{3} \right\} \\
&= \frac{1}{2}(694) + \frac{1}{3} = 347 + \frac{1}{3},
\end{aligned}$$

$$\Psi_1 = \|\omega_1\|_\infty = 0, \qquad \Psi_2 = \|\omega_2\|_\infty = 3, \qquad \text{and} \qquad \Psi_3 = \|\omega_3\|_\infty = 3$$

Using Lemma 3.3, we can compute degree bounds for each λ_i as follows:

$$\deg(\lambda_1) \leq 347 + \frac{1}{3} - 0 = 347 + \frac{1}{3} \tag{3.10}$$

$$\deg(\lambda_2) \leq 347 + \frac{1}{3} - 3 = 344 + \frac{1}{3} \tag{3.11}$$

$$\deg(\lambda_3) \leq 347 + \frac{1}{3} - 3 = 344 + \frac{1}{3} \tag{3.12}$$

When searching for solutions, we would have to compute the norms of $5^{348+345+345} = 5^{1038}$ elements of the form of $\sum_{i=1}^{3} \lambda_i \omega_i$ with polynomials λ_1, λ_2, and λ_3 of degrees bounded as in (3.10), (3.11), and (3.12). On an Intel Xeon CPU E7-8891 v4 with 80 64-bit cores at 2.80GHz, the computation did not end for 4 days, thus we terminated the computation.

As seen in Example 3.1.2, even for small n, g, q and $\deg c$, the search space and the height of the solution α can be large. In the following section, we provide a detailed complexity analysis. Then in Section 3.2, we introduce a new algorithm that uses compact representations to enumerate elements in the search space and to represent solutions of norm equations (2.7).

3.1.2 Complexity analysis

In this section, we carefully analyze the complexity of Algorithm 10 that uses the approach of Gaál and Pohst from [21], described in Section 3.1. All computational costs are expressed in the number of bit operations. Note that using the algorithms in [23], an operation in k takes $O((\log q)^{1+\varepsilon}) = O(q^{\varepsilon})$ bit operations.

First, we count the number of elements for which we compute the norms in Algorithm 10.

Lemma 3.4. *In Algorithm 10 the number of elements whose norm is computed in Step 5 is*

$$q^{\sum_{i=1}^{n}(\lfloor \Theta - \Psi_i \rfloor + 1)},$$

with Θ and Ψ_i as given in Lemma 3.3.

Proof. Algorithm 10 computes the norm of each α_{temp} whose basis coefficients λ_i as given in (3.1) satisfy the degree bounds in Lemma 3.3. Note that in $k[x]$, there are q^{m+1} polynomials which have degree less than or equal to $m \in \mathbb{Z}^+$. Thus, the number of choices for each λ_i for $1 \leq i \leq n$ is,

$$q^{\lfloor \Theta - \Psi_i \rfloor + 1}.$$

Forming all combinations, the total number of elements for which we compute norms in the algorithm is

$$\prod_{i=1}^{n} q^{\lfloor \Theta - \Psi_i \rfloor + 1},$$

which is equivalent to the expression stated in this lemma. □

Combining the results of Lemmas 2.36 and 3.4, we have the cost of Algorithm 10 as follows.

Theorem 3.5. *Let $T = r2^{r(r-1)/4-1}R_F + \frac{1}{n}\deg c$, $d_T = \max\{T, \lceil \frac{2g-1}{n} \rceil\}$, where r is the unit*

rank of O_F. Then Algorithm 10 can solve Equation (2.7) in

$$O\left(2^{nTq^{\varepsilon}}q^{\varepsilon}(n^3 d_T^{1+\epsilon} + (\deg c)^{\omega})\right)$$

bit operations, when n, g, q and $\deg c \to \infty$.

Proof. Consider α_{temp} defined in Step 4 of the algorithm. Due to the construction, we know $\mathbf{h}_{\mathcal{B}}(\alpha_{temp}) \leq \Theta$ with Θ defined in Lemma 3.2. By Corollary 2.30, we have

$$\Theta \leq \frac{r}{2}2^{r(r-1)/4}R_F + \frac{1}{n}\deg c.$$

Thus, computing the norm of each α_{temp} can be done in time $O(n^3 d_T^{1+\epsilon}q^{\varepsilon})$ by Lemma 2.36. By Lemma 3.4, we compute the norm of $q^{\sum_{i=1}^{n}(\lfloor\Theta-\Psi_i\rfloor+1)} \leq q^{n\Theta} = 2^{n\Theta\log q}$ elements. In addition, the cost of testing associateness is as in (2.26), which is $O((\deg c)^{\omega}q^{\varepsilon})$. Let $\mathcal{R}$ be the set of non-associate solutions of the norm equation. We perform Algorithm 1 $|\mathcal{R}|^2$ times. $|\mathcal{R}|$ is less than $q^{n\Theta}$, and in fact, $|\mathcal{R}|^2 \leq q^{n\Theta}$ (for details, see Remark after Lemma 3.8) $\square$

Let α be a solution of (2.7) obtained from Algorithm 10. The size of α is in the notion of the height with respect to the reduced basis $\mathcal{B}$. From Lemma 3.3, α satisfies the following bound on its height,

$$\mathbf{h}_{\mathcal{B}}(\alpha) \leq \Theta. \tag{3.13}$$

Now we discuss asymptotic complexity of Algorithm 10 in different settings; one of n, g, $\deg(c)$, q goes to infinity and the other variables are fixed.

When n goes to infinity and $\deg c$, g, q are fixed, T in the theorem is $O(r2^{(r^2-r)/4}) = O(n2^{n^2/4})$, and $d_T = O(T)$. Thus, the cost of Algorithm 10 is

$$O\left(n^4 2^{n^2 R_F 2^{n^2/4}q^{\varepsilon}+n^2/4}\right) = 2^{O(n^2 2^{n^2/4})}, \tag{3.14}$$

which is doubly exponential as a function of n. We have a bound on R_F from (2.17),

$$R_F \leq (\sqrt{q}+1)^{2g}. \tag{3.15}$$

Thus, the cost of Algorithm 10 when $h_{O_F} = 1$ is

$$O\left(n^4 2^{(n^2 q^g 2^{n^2/4}+g)q^\varepsilon + n^2/4}\right). \tag{3.16}$$

The cost of Algorithm 10, when n goes to infinity, is doubly exponential in n, mainly because the number of elements in the search space is already doubly exponential in n. In this setting, $\mathbf{h}_{\mathcal{B}}(\alpha)$ is superexponential in n.

When g goes to infinity, and the other variables are fixed, the asymptotic complexity of Algorithm 10 is

$$O\left(2^{n^2 R_F 2^{n^2/4} q^\varepsilon} R_F^{1+\varepsilon}\right).$$

Using the bound in (3.15), the asymptotic complexity is

$$O\left(g q^\varepsilon 2^{(n^2 q^g 2^{n^2/4}+g)q^\varepsilon}\right) = 2^{O(q^g)},$$

which is doubly exponential as a function of g. This is also mainly because the number of elements in the search space is already doubly exponential in g. In this setting, $\mathbf{h}_{\mathcal{B}}(\alpha)$ is exponential in g.

When $\deg c$ goes to infinity and the other variables are fixed, we have $q^T = O(q^{\frac{1}{n}\deg c})$. Thus, in this case, the asymptotic complexity of Algorithm 10 is

$$O\left(2^{q^\varepsilon \deg c}(\deg c)^\omega\right), \tag{3.17}$$

which is exponential in $\deg c$, because the number of elements in the search space is exponential in $\deg c$. In this setting, $\mathbf{h}_{\mathcal{B}}(\alpha)$ is polynomial in $\deg c$.

Finally, when q goes to infinity, and the other variables are fixed, the asymptotic complexity of Algorithm 10 is

$$O\left(q^{n^2 R_F 2^{n^2/4} + \frac{1}{n}\deg(c) + \varepsilon}\right), \tag{3.18}$$

which is superexponential in q. By (3.15), the bound is

$$O\left(q^{n^2 q^g 2^{n^2/4} + \frac{1}{n}\deg(c) + \varepsilon}\right) = q^{O(q^g)}. \tag{3.19}$$

The cost of Algorithm 10, when q goes to infinity, is superexponential in q, also because the number of elements in the search space is exponential in q. In this setting, $\mathbf{h}_{\mathcal{B}}(\alpha)$ is polynomial in q.

Table 3.1 shows the summary of the asymptotic complexities of Algorithm 10.

	Algorithm 10
$n \to \infty$	$2^{O\left(n^2 2^{n^2/4}\right)}$
$g \to \infty$	$2^{O(q^g)}$
$\deg c \to \infty$	$O\left(2^{q^\varepsilon \deg c}(\deg c)^\omega\right)$
$q \to \infty$	$q^{O(q^g)}$

Table 3.1: Summary of asymptotic complexity of Algorithm 10

In conclusion, the analysis shows the cost of the algorithm is highly dependent on all 4 variables. The search space given in Lemma 3.4 is doubly exponential in n and g, superexponential in q, and exponential in $\deg(c)$. This is all because of the number of elements in the search space, which is determined by Θ and Ψ_i in Lemma 3.3. While Ψ_i tend to stay small because $\mathcal{B} = \{\omega_i | 1 \leq i \leq n\}$ is a reduced basis, Θ can be large, because $v_{P_\infty}(\varepsilon_j)$ can be large. Thus, it is not efficient to use this algorithm when a system of fundamental units contains huge elements, i.e. the values $v_{P_\infty}(\varepsilon_j)$ are large. In addition, the heights of

the solutions are bounded as in (3.13), which can be superexponential in n, exponential in g, and polynomial in $\deg c$. We decrease the number of elements and the heights by using compact representations in Section 3.2.

3.2 Gaál-Pohst approach with compact representations

This section provides a new algorithm for solving norm equations defined in (2.7), which is inspired by the Gaál-Pohst approach in the previous section and its complexity. The new algorithm uses compact representations to represent elements in the search space instead of the standard representation and to represent solutions α of (2.7). In Section 3.1, we used the standard representation for α as in (3.1) and computed the upper bounds on λ_i in Lemma 3.3. In this section, we compute bounds on $v_P(\alpha)$ for $P \in \mathbb{P}(F)$ and use them to form inputs for Algorithm 4 to compute compact representations.

As we can see in Example 3.1.2 and Section 3.1.2, solving norm equations using Algorithm 10 can take a lot of time even for relatively small q, n, g and $\deg c$, unless F has a system of fundamental units of very small heights. This is because the search space is exponential in the values of the fundamental units at the infinite places. In addition, the solutions of norm equations can also have huge heights. Thus, incorporating compact representations is both computationally efficient and space-efficient.

3.2.1 Algorithm description

In this section, we describe a new exhaustive search algorithm solving norm equations using compact representations.

First, we identify the places $P \in \mathbb{P}(F)$, at which we want bounds on values of α. Given $c \in k[x] \setminus k$, let $S_{c,0}$ be the set of finite places of F at which c has non-zero values, and $S_c = S_{c,0} \cup \mathbb{P}_\infty(F)$. Denote the elements of $S_{c,0}$ by P_i for $1 \leq i \leq |s_{c,0}|$ and the elements of $\mathbb{P}_\infty(F)$ by $P_{\infty,i}$ for $1 \leq i \leq |\mathbb{P}_\infty(F)|$, unless specified otherwise. Any solution α of

equation (2.7) is an S_c-unit. That means that it is sufficient to compute bounds on $v_P(\alpha)$ for $P \in \mathbb{P}_\infty(F) \cup S_{c,0}$ due to the definition in (2.15).

Bounds on $v_{P_\infty}(\alpha)$, for $P_\infty \in \mathbb{P}_\infty(F)$ can be derived from the equality in (3.4) as follows.

Lemma 3.6. *For $1 \leq i \leq |\mathbb{P}_\infty(F)|$, let*

$$\Theta'_i = \sum_{j=1}^{r} \frac{1}{2} |v_{P_{\infty,i}}(\varepsilon_j)|.$$

Then for each $P_{\infty,i} \in \mathbb{P}_\infty(F)$, $v_{P_{\infty,i}}(\alpha)$ is bounded as below.

$$-\Theta'_i - \frac{e_{P_\infty|p_\infty}}{n} \deg(c) \leq v_{P_\infty}(\alpha) \leq \Theta'_i - \frac{e_{P_\infty|p_\infty}}{n} \deg(c).$$

Proof. The equality in (3.4) can be directly transformed to the result, using the fact $-1/2 \leq x_{\alpha_0,j} - \lfloor x_{\alpha_0,j} \rceil \leq 1/2$ for $1 \leq j \leq r$. □

Now we consider the bounds on $v_P(\alpha)$ for the finite places P. For any place $P \in S_{c,0}$, the value $v_P(\alpha)$ is bounded as follows.

Lemma 3.7. *For any place $P \in S_{c,0}$, a solution $\alpha \in O_F$ of equation (2.7) satisfies*

$$0 \leq v_P(\alpha) \leq v_P(c).$$

Proof. Since c is the norm of $\alpha \in O_F$ up to a constant, $v_P(\mathrm{Norm}_{F/k(x)}(\alpha)) = v_P(\prod_{i=1}^{n} \sigma_i(\alpha)) = v_P(c)$. Let F_{split} be a splitting field of c. Then F_{split}/F is a finite extension and $F_{split}/k(x)$ is a Galois extension. Let Q be a finite place of F_{split} above $P \in S_{c,0}$ with ramification index $e_{Q|P}$. Then we have

$$v_Q(\alpha) = e_{Q|P} v_P(\alpha) \quad \text{and} \quad v_Q(\sigma_i(\alpha)) = v_{\sigma_i^{-1}(Q)}(\alpha).$$

The latter equality is by [42, Lemma 3.5.2]. Thus,

$$\begin{aligned} v_P(c) &= v_P(\mathrm{Norm}_{F/k(x)}(\alpha)) = v_P(\prod_{i=1}^{n} \sigma_i(\alpha)) \\ &= \frac{\sum_{i=1}^{n} v_Q(\sigma_i(\alpha))}{e_{Q|P}} = \frac{\sum_{i=1}^{n} v_{\sigma_i^{-1}(Q)}(\alpha)}{e_{Q|P}} \\ &\geq \frac{v_Q(\alpha)}{e_{Q|P}} = v_P(\alpha) \end{aligned}$$

The inequality holds because $\alpha \in O_F$. □

With these bounds, we form inputs for Algorithm 4 to compute compact representations. Algorithm 4 takes two inputs: an ideal I, and an integer vector V_∞. We use the bounds on $v_P(\alpha)$ to form I, and the bounds on $v_{P_\infty}(\alpha)$ to form V_∞ as follows.

Let $\{\mathfrak{p}_i | 1 \leq i \leq |S_{c,0}|\}$ be the set of the prime ideals corresponding to the places in $S_{c,0}$. Consider the ideal of the following form,

$$I = \prod_{i=1}^{|S_{c,0}|} \mathfrak{p}_i^{v_i}, \tag{3.20}$$

where $0 \leq v_i \leq v_{P_i}(c)$.

Then consider the vector of the form

$$V_\infty = \begin{bmatrix} v_{\infty,1} & \dots & v_{\infty,|\mathbb{P}_\infty(F)|} \end{bmatrix}, \tag{3.21}$$

where each $v_{\infty,i}$ satisfies the inequality in Lemma 3.6,

$$-\Theta_i' - \frac{e_{P_{\infty,i}|p_\infty}}{n} \deg(c) \leq v_{\infty,i} \leq \Theta_i' - \frac{e_{P_{\infty,i}|p_\infty}}{n} \deg(c). \tag{3.22}$$

Each pair I and V_∞ from (3.20) and (3.21) can be used as inputs to Algorithm 4. However, it is not necessary to compute compact representations for all possible pairs. Note that the inputs to compute a compact representation of α are the principal ideal αO_F generated by

α and the vector $val_\infty(\alpha)$ of values of α at the infinite places of F. Consider the divisor of α

$$\mathrm{div}(\alpha) = \sum_{i=1}^{|S_{c,0}|} v_{P_i}(\alpha) P_i + \sum_{j=1}^{|\mathbb{P}_\infty(F)|} v_{P_{\infty,j}}(\alpha) P_{\infty,j}.$$

Since $\mathrm{div}(\alpha)$ is a principal divisor, its degree is zero.

$$\sum_{i=1}^{|S_{c,0}|} v_i \deg P_i + \sum_{j=1}^{|\mathbb{P}_\infty(F)|} v_{\infty,j} \deg P_{\infty,j} = 0. \tag{3.23}$$

Satisfying (3.23) does not mean that the divisor is principal. However, by checking this equality, we eliminate many I and V_∞ before computing compact representations.

Once $\mathbf{t} = \mathrm{CompRep}(I, V_\infty)$ is obtained, we test that $\mathbf{t}$ is in fact in O_F, and check its norm using Algorithm 7. Checking whether or not $\mathbf{t}$ is in O_F can be done by computing the values of $\mathbf{t}$ at $P \in S_{c,0}$ using Algorithm 8 and ensuring that they are positive. Finally, for every new solution, we remove associate solutions using Algorithm 9, and only collect non-associate solutions. Algorithm 11 shows the whole process we described above.

Example 3.2.1 is an example of Algorithm 11, solving the norm equation in Example 3.1.2.

Example 3.2.1. This example deals with the same norm equation in Example 3.1.2. Let F, f, and c be as in Example 3.1.2, $F/\mathbb{F}_5(x)$ is a finite extension of degree $n = 3$ by

$$f(t) = t^3 + (4x^3 + 3x^2 + 1)t^2 + (3x^3 + 4x^2 + 4x + 2)t + 2x^3 + x,$$

and $c = x+4$. F has two infinite places $\{P_{\infty,1}, P_{\infty,2}\}$ of F with ramification indices $e_{P_{\infty,1}|p_\infty} = e_{P_{\infty,2}|p_\infty} = 1$. We have 2 prime ideals that are the factors of cO_F; $\mathfrak{p}_1$ and $\mathfrak{p}_2$ that correspond to the finite places P_1 and P_2, and $v_{P_1}(c) = v_{P_2}(c) = 1$.

Algorithm 11 NormEquation-GaalPohstCR

Input: $c \in k[x] \setminus \{0\}$, a reduced basis $\mathcal{B} = \{\omega_i | 1 \leq i \leq n\}$ of O_F, a unit value matrix $M_{\mathbb{P}_\infty(F)}$

Output: A set $\mathcal{R}$ of all non-associate solutions of the equation (2.7) in compact representation

$\mathcal{R} \longleftarrow \emptyset$
$S_{c,0} \longleftarrow \{P \in \mathbb{P}_0(F) | v_P(c) \neq 0\}$
$v_l \longleftarrow \left[\sum_{i=1}^{r} \frac{1}{2}|(M_{\mathbb{P}_\infty(F)})_{i,1}| \quad \dots \quad \sum_{i=1}^{r} \frac{1}{2}|(M_{\mathbb{P}_\infty(F)})_{i,|\mathbb{P}_\infty(F)|}|\right]$
$v_c \longleftarrow \left[\frac{e_{P_{\infty,1}|p_\infty}}{n} \deg(c) \quad \dots \quad \frac{e_{P_{\infty,|\mathbb{P}_\infty(F)||p_\infty}}}{n} \deg(c)\right]$
for $I = \prod_{i=1}^{|S_{c,0}|} \mathfrak{p}_i^{v_i}$, of $\mathrm{Norm}(I)/c \in k$, where $0 \leq v_i \leq v_P(c)$ **do**
  for $V_\infty = \left[v_{\infty,1} \quad \dots \quad v_{\infty,|\mathbb{P}_\infty(F)|}\right]$ where $-(v_l)_i - (v_c)_i \leq v_{\infty,i} \leq (v_l)_i - (v_c)_i$ **do**
    if $\sum_{i=1}^{|S_{c,0}|} v_i \deg P_i + \sum_{j=1}^{|\mathbb{P}_\infty(F)|} v_{\infty,j} \deg P_{\infty,j} = 0$ **then**
      $\mathbf{t} \longleftarrow \mathrm{CompRep}(I, V_\infty)$
      if $\mathbf{t} \in O_F$ and $\mathrm{NormCR}(\mathbf{t})/c \in k^\times$ **then**
        if not (IsAssociateCR($\mathbf{t}$, $\mathbf{t}_\alpha$)) for all $\mathbf{t}_\alpha \in \mathcal{R}$ **then**
          $\mathcal{R} \longleftarrow \mathcal{R} \cup \{\mathbf{t}\}$
          break V_∞
        end if
      end if
    end if
  end for
end for
return $\mathcal{R}$

As in Example 3.1.2, we have

$$M_{\mathbb{P}_\infty(F)} = \mathrm{SValMat}(\mathbb{P}_\infty(F)) = \begin{bmatrix} 694 & -347 \end{bmatrix}.$$

Using $M_{P_\infty(F)}$, we compute the bounds in Lemmas 3.6 and 3.7 as follows. For $P_{\infty,1}$,

$$\sum_{j=1}^{r} \left(-\frac{1}{2} |v_{P_{\infty,1}}(\varepsilon_j)| \right) - \frac{e_{P_{\infty,1}|p_\infty}}{n} \deg(c) = -\frac{1}{2}(694) - \frac{1}{3} = -347 - \frac{1}{3}, \quad \text{and}$$

$$\sum_{j=1}^{r} \left(\frac{1}{2} |v_{P_{\infty,1}}(\varepsilon_j)| \right) - \frac{e_{P_{\infty,1}|p_\infty}}{n} \deg(c) = \frac{1}{2}(694) - \frac{1}{3} = 347 - \frac{1}{3}.$$

For $P_{\infty,2}$,

$$\sum_{j=1}^{r} \left(-\frac{1}{2} |v_{P_{\infty,2}}(\varepsilon_j)| \right) - \frac{e_{P_{\infty,2}|p_\infty}}{n} \deg(c) = -\frac{1}{2}(347) - \frac{1}{3} = -\frac{347}{2} - \frac{1}{3} = -173 - \frac{5}{6}, \quad \text{and}$$

$$\sum_{j=1}^{r} \left(\frac{1}{2} |v_{P_{\infty,2}}(\varepsilon_j)| \right) - \frac{e_{P_{\infty,2}|p_\infty}}{n} \deg(c) = \frac{1}{2}(694) - \frac{1}{3} = \frac{347}{2} - \frac{1}{3} = 173 + \frac{1}{6}.$$

With the values of c at the finite places, we have four ideals,

$$I_1 = 1 \cdot O_F \qquad I_2 = \mathfrak{p}_1, \qquad I_3 = \mathfrak{p}_2, \qquad I_4 = \mathfrak{p}_1\mathfrak{p}_2.$$

The number of all possible V_∞ is $(347{\cdot}2){\cdot}(173{\cdot}2{+}1) = 240818$ which is much smaller than 5^{1038} in Example 3.1.2. By testing the equality in (3.23) and searching for solutions as in Algorithm 11, we get one solution in compact representation, $\mathfrak{t}$ =CompRep$\left(\mathfrak{p}_1, \begin{bmatrix} -317 & 158 \end{bmatrix}\right)$, that is

$\mathbf{t} = (\mu, \beta_1, \beta_2, \ldots, \beta_9)$ where

$$
\begin{aligned}
\mu =& y + 4 \\
\beta_1 =& \beta_2 = \beta_9 = 1,\ \beta_3 = 4 \\
\beta_4 =& (3x^2 + 3x)y^2 + (2x^5 + x^4 + 4x^3 + 4x^2 + 3x + 1)y + 3x^5 + 4x^4 + 3x^3 + x^2 + 3x + 2 \\
\beta_5 =& \frac{3x^5 + 4x^4 + 3x^3 + 2x^2 + 3x + 1}{x^8 + 2x^7 + 2x^6 + 4x^5 + x^4 + 3x^3 + 3x^2 + x + 4} y^2 \\
&+ \frac{2x^8 + 4x^6 + 3x^5 + x^4 + 2x^3 + 3x + 1}{x^8 + 2x^7 + 2x^6 + 4x^5 + x^4 + 3x^3 + 3x^2 + x + 4} y \\
&+ \frac{x^8 + 4x^7 + 3x^6 + x^5 + 2x^4 + x^3 + 4x^2 + 4x + 3}{x^8 + 2x^7 + 2x^6 + 4x^5 + x^4 + 3x^3 + 3x^2 + x + 4} \\
\beta_6 =& \frac{x^4 + 2x^3 + 2x^2 + 2}{x^7 + 2x^5 + 3x^4 + 2x^2 + x + 1} y^2 + \frac{4x^7 + x^6 + 4x^5 + 2x^4 + 3x^3 + x^2 + 2x + 1}{x^7 + 2x^5 + 3x^4 + 2x^2 + x + 1} y \\
&+ \frac{3x^7 + 2x^6 + 4x^5 + 2x^4 + 2x^2 + 2x + 4}{x^7 + 2x^5 + 3x^4 + 2x^2 + x + 1} \\
\beta_7 =& \frac{2x^4 + x^3 + x^2 + x + 4}{x^8 + 2x^7 + x^6 + x^4 + x^3 + 4} y^2 + \frac{3x^7 + 3x^5 + 2x^3 + 4x + 4}{x^8 + 2x^7 + x^6 + x^4 + x^3 + 4} y \\
&+ \frac{4x^8 + 3x^7 + 2x^6 + 3x^5 + x^4 + x^3 + 2x^2 + 3x + 4}{x^8 + 2x^7 + x^6 + x^4 + x^3 + 4} \\
\beta_8 =& \frac{4x^4 + 3x^3 + 2x^2 + 3x + 2}{x^8 + 4x^7 + 3x^6 + 2x^4 + 2x^3 + 4x^2 + 4x + 4} y^2 + \frac{x^5 + 4x^4 + 4x^3 + x^2 + 3}{x^6 + 4x^5 + 3x^3 + 2x^2 + 3x + 3} y \\
&+ \frac{x^5 + x^4 + 4x^3 + 3x^2 + 3x + 3}{x^6 + 4x^5 + 3x^3 + 2x^2 + 3x + 3}.
\end{aligned}
$$

Thus $\{\mathbf{t}\}$ is the set of non-associate solution of equation (2.7) with $c = x + 4$. The entire process took 114.830 CPU seconds on an Intel Xeon CPU E7-8891 v4 with 80 64-bit cores at 2.80GHz.

While the same machine was not able to solve the norm equation using Algorithm 10 for 4 days, it only took 114.830 CPU seconds solving it using Algorithm 11. In the following section, we provide a detailed complexity analysis of Algorithm 11 and compare it to the complexity of Algorithm 10.

3.2.2 Complexity analysis

In this section, we analyze the complexity of Algorithm 11. For Algorithm 11, we have an assumption that F has at least one infinite place of degree 1, which is necessary for computing compact representations. All computational costs are expressed in terms of the number of bit operations. Note that using the algorithms in [23], an operation in k takes $O((\log q)^{1+\varepsilon}) = O(q^{\varepsilon})$ bit operations.

First, we count the number of I and V_∞. The number of I of the form (3.20) is determined by $v_P(c)$ as follows.

Lemma 3.8. *Let $c \in k[x] \setminus k$, and $cO_F = \prod_{i=1}^{|S_{c,0}|} \mathfrak{p}_i^{v_{P_i}(c)}$, where $\mathfrak{p}_i$ are distinct prime ideals of O_F that are corresponding to places P_i in $S_{c,0}$. Then the number of I of the form (3.20) is*

$$\prod_{i=1}^{|S_{c,0}|} (v_{P_i}(c) + 1).$$

Proof. Each I is of the form of (3.20). Since each v_i can range from 0 to $v_{P_i}(c)$ for $1 \leq i \leq |S_{c,0}|$, the number of such I is equal to the number of exponent vectors of the form $\begin{bmatrix} v_1 & v_2 & \dots & v_{|S_{c,0}|} \end{bmatrix}^T$, where $0 \leq v_i \leq v_{P_i}(c)$ for $1 \leq i \leq |S_{c,0}|$. Therefore, the number of I is as stated in this lemma. □

Remark. From the lemma above, the number of non-associate solutions of any norm equation (2.20) is bounded by $\prod_{i=1}^{|S_{c,0}|}(v_{P_i}(c) + 1)$.

Remark. If cO_F is irreducible, the least expensive case is that there is only one place P of F above c, which is unramified. Then we have $|S_{c,0}| = 1$, $v_P(c) = 1$, and the number of matrix equations is 2.

Remark. If cO_F is irreducible, the most expensive case is that c splits into linear polynomials, and each linear factor splits completely in F. Then the dimension of each matrix is $n \deg(c) \times (n \deg(c) + r)$, and the number of matrix equations is $2^{n \deg(c)}$.

The number of V_∞ of the form (3.21) can be counted similarly. Each $v_{\infty,i}$ is in the interval

(3.22), thus the number of V_∞ is

$$\prod_{j=1}^{r+1}(2\Theta'_j+1),$$

where $\Theta'_j = \sum_{i=1}^{r}\left(\frac{1}{2}|v_{P_{\infty,j}}(\varepsilon_i)|\right)$. Thus, the total number of pairs I and V_∞ is

$$\left(\prod_{i=1}^{|S_{c,0}|}(v_P(c)+1)\right)\left(\prod_{j=1}^{r+1}(2\Theta'_j+1)\right).$$

For each I, we compute its norm in Step 5. The cost of computing the norm of I in HNF representation is $O\left((n^{2+\varepsilon}\deg(c)+(n\deg(c))^{1+\varepsilon})\,q^{\varepsilon}\right)$ as in Lemma 2.37 The cost of computing the norm of I is negligible compared to the cost of the rest of the algorithm including computing compact representations and their norms. The following lemma provides the number of compact representations computed in the algorithm.

Lemma 3.9. *In Algorithm 11, the number of compact representations computed is bounded by*

$$\left(\prod_{i=1}^{|S_{c,0}|}(v_P(c)+1)\right)\left(\prod_{j=1}^{r}(2\Theta'_j+1)\right),$$

where $\Theta'_j = \sum_{i=1}^{r}\left(\frac{1}{2}|v_{P_{\infty,j}}(\varepsilon_i)|\right)$

Proof. We compute compact representations after checking the degree given in (3.23). That process drops one factor of $2\Theta'_j+1$. That is because, when given I and a vector $V'_\infty = \begin{bmatrix} v_{\infty,1} & \dots & v_{\infty,|\mathbb{P}_\infty(F)|-1}\end{bmatrix}$, there is at most one $v_{\infty,|\mathbb{P}_\infty(F)|}\in\mathbb{Z}$ that makes $V_\infty = \begin{bmatrix} v_{\infty,1} & \dots & v_{\infty,|\mathbb{P}_\infty(F)|}\end{bmatrix}$ satisfy (3.23). □

Using Lemma 2.42, computing each compact representation costs

$$O\left(gq^{\varepsilon}\left(\mathsf{RR}\left(n\deg(c)+g\right)+\log(\max_i\Theta_i+n+g)\mathsf{RR}\left(n^2+ng\right)\right)\right) \tag{3.24}$$

bit operations. The following lemma shows the height bounds of each component of a solution in compact representation obtained from Algorithm 11.

Lemma 3.10. *Let* $\mathbf{t} = (\mu, \beta_1, \ldots, \beta_l)$ *be the compact representation of* α *that is computed by CompRep(A,* $v_{\alpha,\infty} - v_0$*) in Step 8 of Algorithm 11. Then we have*

$$\begin{aligned} l &= O\left(\log \left\| M^T_{S_c,\infty} \right\|_\infty\right), \\ \mathbf{h}(\mu) &= O(\mathbf{h}(\mathrm{div}(cO_F)) + g + n^2 C_f) = O(n \deg c + g + n^2 C_f), \text{ and} \\ \mathbf{h}(\beta_i) &= O(n^2 + ng + n^2 C_f) \text{ for } 1 \le i \le l. \end{aligned}$$

Proof. Due to the construction of I, $\mathrm{div}(I) \le \mathrm{div}(cO_F)$. Thus, $\mathbf{h}(\mathrm{div}(I)) \le \mathbf{h}(\mathrm{div}(cO_F))$. The result follows from Lemma 2.44. □

Thus, the cost of Algorithm 11 is as follows.

Theorem 3.11. *With* $T = r2^{r(r-1)/4-1}R_F$*, Algorithm 11 can solve Equation (2.7) in*

$$O\Bigg(\big(g\mathsf{RR}\left(n\deg(c)+g\right) + g\log(T+n+g)\mathsf{RR}\left(n^2+ng\right) \\ +n^{5+\varepsilon}(R_F \deg c)^{1+\varepsilon} + (\deg c)^{\omega}\big) q^{\varepsilon}(2T+1)^r \left(\prod_{P\in S_{c,0}} (v_P(c)+1)\right)\Bigg)$$

bit operations, when n*,* g*,* q *and* $\deg c \to \infty$*.*

Proof. From Theorem 3.9, we have

$$\left(\prod_{P\in S_{c,0}} (v_P(c)+1)\right)\left(\prod_{j=1}^{r}(2\Theta'_j+1)\right)$$

compact representations computed in the algorithm. With the cost of computing compact representation in (3.24), computing each compact representation takes

$$O\left(gq^{\varepsilon}\left(\mathsf{RR}\left(n\deg(c)+g\right) + \log(\max_i \Theta_i + n + g)\mathsf{RR}\left(n^2+ng\right)\right)\right)$$

bit operations. By Corollary 2.30, we have

$$\Theta_i \leq \Theta \leq \frac{r}{2} 2^{r(r-1)/4} R_F,$$

for $1 \leq i \leq |\mathbb{P}_\infty(F)|$. The cost of testing whether or not a compact representation is in O_F is done using Algorithm 8. The cost is as in (2.26), $O((\deg c)^\omega q^\varepsilon)$. The cost of computing the norm of a compact representation using Algorithm 7 is $O(n^{5+\varepsilon} R_F (\deg c)^{1+\varepsilon} q^\varepsilon)$ combining the results of Lemma 3.10 and Lemma 2.48 Finally, Algorithm 9 tests whether or not two compact representations are associate. The cost of Algorithm 9 is also dominated by the cost of Algorithm 8, which is as in (2.26), $O((\deg c)^\omega q^\varepsilon)$. □

Now we discuss the asymptotic complexity of Algorithm 11 in different settings; one of n, g, $\deg c$, q goes to infinity, and the other variables are fixed.

In order to simplify the complexity analysis, we first bound $\prod_{P \in S_{c,0}} (v_P(c) + 1)$. In Theorem 3.11, the complexity of Algorithm 11 has a factor of $\prod_{P \in S_{c,0}} (v_P(c) + 1)$. This number varies greatly depending on the inputs. The minimum is when c and cO_F are irreducible and the place P above cO_F is unramified, so $v_P(c) = 1$. The maximum is when c splits into linear factors c_i for $1 \leq i \leq \deg c$, and each $c_i O_F$ has n places above it. In this case, $v_P(c) = 1$ for all $P \in S_{c,0}$ and $|S_{c,0}| = n \deg c$. Thus,

$$2 \leq \prod_{P \in S_{c,0}} (v_P(c) + 1) \leq 2^{n \deg c}. \tag{3.25}$$

When n goes to infinity, and the other variables are fixed, T in Theorem 3.11 is $O(r 2^{r(r-1)/4-1} R_F) = O(n 2^{n^2/4} R_F)$, and $d_T = O(T)$. The worst case is when $\prod_{P \in S_{c,0}} (v_P(c) + 1)$ is maximum. In this case, $|S_c| = O(n)$ and $\prod_{P \in S_{c,0}} (v_P(c) + 1) = O(2^{n \deg c})$. Using the cost of computing Riemann-Roch space in (2.33), $\mathsf{RR}(n^2 + ng)$ can be simplified to $O(n^9)$ in this case. Thus, the cost of Algorithm 11 is

$$O\left(n^{11} 2^{n^3/4 + n \deg c + n^{1+\varepsilon}} R_F^n\right) = 2^{n^3(1+o(1))}.$$

which is superexponential in n, because the number of compact representations computed in the algorithm is superexponential in n. It is far less than the complexity of Algorithm 10 in (3.14), which is doubly exponential in n. The decrease is mainly because of the smaller search space of Algorithm 11. The number of elements in the search space of Algorithm 11 in this setting is $O(2^{n^3/4+n\deg c+n^{1+\varepsilon}}R_F^n)$. This replaces the dominant factor, $O(2^{n^2 2^{n^2/4}}R_F q^{\varepsilon})$, of Algorithm 10, which is the number of elements in its search space. In addition, in consequence of using compact representations, Algorithm 11 has a better asymptotic complexity for checking whether or not each element in the search space is a solution of (2.7). The number of bit operations needed to compute the norm of a compact representation in Algorithm 11 is $O(n^{11})$, while Algorithm 10 computes the norm of an element in $O(n^4 2^{n^2/4})$ bit operations. Since we have (3.15), the cost of Algorithm 11 is

$$O\left(n^{11}2^{n^3/4+n\deg c+n^{1+\varepsilon}+gnq^{\varepsilon}}\right).$$

Algorithm 11 works slightly faster when c and cO_F are irreducible, in which case we have $\prod_{P\in S_{c,0}}(v_P(c)+1)=2$ as in (3.25), and the asymptotic complexity is

$$O\left(n^{11}2^{n^3/4+n^{1+\varepsilon}}R_F^n\right).$$

Though it is still superexponential in n, it has one less factor of 2^n. In this setting, similar to the bound in (2.36), $\mathbf{h}(\mathrm{div}(cO_F))=O(n)$. By Proposition 2.29 and Lemma 3.2, the size of a solution in compact representation in Algorithm 11 is polynomial in n. It is smaller than Algorithm 10's, because a solution α obtained from Algorithm 10 satisfies $\mathbf{h}_{\mathcal{B}}(\alpha)=O(n2^{n^2/4}R_F)$.

When g goes to infinity and the other variables are fixed, T in Theorem 3.11 is $O(R_F)$, and $d_T=O(T)$. Using the cost of computing Riemann-Roch space in (2.33), $\mathsf{RR}\,(n^2+ng)$ can be simplified to $O(g^{2+\varepsilon})$ in this case because $C_f=O(g)$. Thus, the cost of Algorithm

11 is

$$O\left(g^{3+\varepsilon} R_F^{n+1+\varepsilon}\right).$$

We have $R_F^{\varepsilon} = O(g)$ from (3.15). Thus, the cost of Algorithm 11 is

$$O\left(g^{4+\varepsilon} 2^{g(n+1)q^{\varepsilon}}\right) = 2^{O(g)},$$

which is exponential in g. The exponential factor $2^{gnq^{\varepsilon}}$ is from the number of elements in the algorithm's search space. The asymptotic complexity above is much smaller than Algorithm 10's because the number of elements in the search space of Algorithm 10 is $O(2^{n^2 q^{g+\varepsilon} 2^{n^2/4}})$, which is already doubly exponential in g. In this setting, a solution in compact representation can be written in polynomial size in g, while the height of a solution in the standard representation with respect to a reduced basis $\mathcal{B}$ is $O(q^g)$.

Now we consider the case that $\deg c$ goes to infinity, and the other variables are fixed. The worst case is the same as before, namely when $|S_c| = O(\deg c)$. Using the cost of computing Riemann-Roch space given in (2.33), $\mathsf{RR}\left(n \deg c + g\right)$ can be simplified to $O((\deg c)^2)$ The asymptotic complexity of Algorithm 11 is

$$O\left(2^{n \deg c} (\deg c)^{\omega}\right),$$

which is similar to the asymptotic complexity of Algorithm 10 given in (3.17). While the number of elements in the search space in Algorithm 10 is exponential in $\deg c$ regardless of the number of primes above cO_F, the number of compact representations computed in Algorithm 11 can be less than $O(2^{n \deg c})$. For example, when c and cO_F are irreducible, the complexity of Algorithm 11 is only $O\left((\deg c)^{\omega}\right)$, which is polynomial in $\deg c$. In this setting, similarly bounded as in (2.36), $\mathbf{h}(\mathrm{div}(cO_F)) = O(\deg c)$. Thus, a solution in compact representation can be written in polynomial size in $\deg c$.

Finally, when q goes to infinity, and the other variables are fixed, the asymptotic com-

plexity of Algorithm 11 is

$$O(R_F^{n+\varepsilon} q^{\varepsilon}).$$

Using the bound in (3.15),

$$O(q^{gn+\varepsilon}), \tag{3.26}$$

which is a polynomial in q, and is clearly smaller than (3.19), which is superexponential in q. The superexponential factor $q^{n^2 q^g 2^{n^2/4}}$ of Algorithm 10 comes from the number of elements in the search space. Algorithm 11 successfully removed the superexponential factor because the number of elements in the search space of Algorithm 11 is polynomial in q, namely $O(q^{gn})$. In this setting, the number of field elements required to represent a solution in compact representation is polynomial in $\log q$.

Table 3.2 shows a summary of an asymptotic complexity comparison of Algorithm 10 and Algorithm 11.

In conclusion, the complexity of Algorithm 11 is smaller than the complexity of Algorithm 10 asymptotically in variables, n, g, and q. The asymptotic complexity of Algorithm 11 when $\deg c$ goes to infinity is exponential in $\deg c$, which is similar to Algorithm 10. The search space of Algorithm 11 can be as small as polynomial in n when c and cO_F are irreducible, while the search space of Algorithm 10 does not depend on the number of factors of c. In addition, the size of a solution from Algorithm 11 is polynomial in n, g, $\deg c$, and $\log q$. This is much smaller compared to the height of a solution from Algorithm 10, which can be as big as superexponential in n, exponential in g, polynomial in $\deg c$ and q.

Although it is faster than the existing algorithm and it represents solutions in smaller sizes successfully, Algorithm 11 is still an exhaustive search algorithm searching all elements in its search space. In the following chapter, we introduce a new algorithm for solving norm equations more efficiently by searching ideals instead of elements and using compact representations.

Algorithm 10	Algorithm 11
$n \to \infty$	
$2^{O(n^2 2^{n^2/4})}$	$2^{n^3(1+o(1))}$
$g \to \infty$	
$2^{O(q^g)}$	$2^{O(g)}$
$\deg c \to \infty$	
$O\left(2^{q^\varepsilon \deg c}(\deg c)^\omega\right)$	$O(2^{n \deg c}(\deg c)^\omega)$
$\deg c \to \infty$, c irreducible (best)	
$O\left(2^{q^\varepsilon \deg c}(\deg c)^\omega\right)$	$O((\deg c)^\omega)$
$q \to \infty$	
$q^{O(q^g)}$	$O(q^{gn+\varepsilon})$

Table 3.2: Summary of asymptotic complexity of Algorithm 10 and Algorithm 11

Chapter 4

Solving norm equations: Principal ideal generator approach

In this chapter, we describe an approach for solving norm equations (2.7) in O_F that is based on principal ideal testing. While it was known how to use principal ideals of norm c to find elements of F of norm c, no explicit algorithm for solving (2.7) in global function fields using this approach was available. In addition, we incorporate compact representations to represent solutions in shorter forms. We introduce our new algorithm for solving norm equations with this approach and compact representations in Section 4.1.1. A small example is also provided for illustrative purposes. We also provide a variation of the algorithm at the end of the section. Finally, in Section 4.1.2, we provide a detailed complexity analysis of the algorithm.

We keep the same notation as in the previous chapters. Given $c \in k[x] \setminus k$, we consider a norm equation as in Definition 2.20. Our goal is to find all non-associate solutions α of (2.7) in O_F. We consider that $\alpha \in O_F$ is a solution of (2.7) if it satisfies the norm equation up to a non-zero constant, i.e. satisfy $\text{Norm}_{F/k(x)}(\alpha) = c\zeta$ for any constant $\zeta \in k^\times$.

In order to use this approach meaningfully, we suppose there is at least one finite place of F that has a nonzero value at c. That means c is not a constant, $c \in k[x] \setminus k$, so there

exists at least one finite place $P \in \mathbb{P}_0(F)$ such that $v_P(c) \neq 0$. This assumption is reasonable because solving equation (2.7) with a non-zero constant c in O_F is equivalent to finding a unit of O_F, for which there are better algorithms available. We require that F has at least one infinite place of degree 1 to use compact representations.

4.1 Principal ideal approach using compact representation

This section includes the approach of solving (2.7) in the maximal order O_F by finding all principal ideals of norm c up to a non-zero constant. We provide an explicit algorithm in Section 4.1.1 and the complexity of the algorithm in Section 4.1.2.

4.1.1 Algorithm description

In this section, we introduce a new algorithm for solving norm equations by conducting principal ideal tests and using compact representations. Instead of enumerating all elements satisfying huge bounds as the algorithms in Chapter 3, this algorithm enumerates ideals I that divide cO_F and conducts principal ideal tests by solving matrix equations involving a precomputed S-unit value matrix. Using the solutions of the matrix equations, we compute compact representations of solutions of norm equations.

First, we show that it is sufficient to investigate ideals I that divide cO_F to find all principal ideals of O_F of norm c up to a non-zero constant. Consider a principal ideal αO_F of O_F generated by $\alpha \in F$ whose norm is c up to a non-zero constant. Note that such α and αO_F exists if and only if α is a solution of (2.7). In Lemma 4.1 below, we prove that such αO_F divides the ideal cO_F generated by c. This lemma guarantees that we only need to investigate ideals I dividing cO_F to solve (2.7) by finding all principal ideals of O_F of norm c.

Lemma 4.1. *Given a non-zero polynomial c in $k[x] \setminus \{0\}$, let α be an element of O_F with norm c up to non-zero constant. Then the principal ideal αO_F generated by α divides the principal ideal cO_F generated by c.*

Proof. For such α,

$$\alpha O_F = \prod_{i=1}^{l} \mathfrak{p}_i^{v_{P_i}(\alpha)}.$$

By Lemma 3.7, we know that for any finite place $P \in \mathbb{P}_0(F)$, $v_P(\alpha) \leq v_P(c)$. Thus, αO_F divides cO_F. □

The following procedure finds all non-associate solutions of (2.7) due to the above lemma. For each I that divides cO_F, we test two things; if the norm of I is c and if I is principal. I passes both tests if and only if a generator of I is a solution of (2.7).

We first compute the norms of I. For each I whose norm is c up to a non-zero constant, we perform a principal ideal test and compute generators of those ideals that are principal. The generators form a set of non-associate solutions of (2.7). We do not need to check if the norms of the generators are $c\zeta$ for some $\zeta \in k$ because of (2) of Lemma 2.13. Note that we also do not need to test whether any pair of generators is associate because each solution corresponds to a different ideal of O_F.

In order to enumerate all ideals I that divide cO_F, we factor cO_F as follows,

$$cO_F = \prod_{i=1}^{l} \mathfrak{p}_i^{v_{P_i}(c)}, \tag{4.1}$$

where each $\mathfrak{p}_i$ is the O_F-prime ideal corresponding to the finite place P_i of F. Then we perform principal ideal tests on all ideals I of the form

$$I = \prod_{i=1}^{l} \mathfrak{p}_i^{v_i} \tag{4.2}$$

where v_i are integers, $0 \leq v_i < v_{P_i}(c)$ for $1 \leq i \leq l$.

There is an existing principal ideal test, which is an index calculus algorithm that uses Hess's randomized relation search algorithm [25, Algorithm 5.5]. This algorithm finds a factorization of an ideal equivalent to I by searching relations. When I is principal, the algorithm returns a generator in "factored form". The factored form has subexponentially many terms, each of which has a subexponential size in the size of inputs.

In our case, we already know how I factors from how we enumerate I, and we want to compute a compact representation of a generator of I if I is principal. Thus, instead of using the existing algorithm, for each I, we solve a matrix equation to determine whether or not I is principal and also to compute inputs for computing a compact representation.

For each ideal I of norm c up to a non-zero constant, we test if I is principal by solving a matrix equation involving an S-unit value matrix, which is precomputed using Algorithm 2. If I is principal, we compute a generator α_I of I, which is a solution of (2.7). Since a generator of α_I can have a huge size in standard representation, we use compact representations to represent α_I.

In order to form a matrix equation for the principal ideal test, we consider the following. Given $c \in k[x] \setminus k$, let $S_{c,0} = \{P_1, \ldots, P_{|S_{c,0}|}\}$ be the set of finite places of F at which c has non-zero values, and $S_c = S_{c,0} \cup \mathbb{P}_\infty(F)$. We know that any solution α of equation (2.7) is an S_c-unit; thus, α can be written as a power product of a system of fundamental S_c-units, $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_{r_{S_c}}\}$,

$$\alpha = \prod_{i=1}^{r_{S_c}} \epsilon_i^{x_i}, \tag{4.3}$$

where $x_i \in \mathbb{Z}$. In addition, for any ideal I dividing cO_F, we can write

$$I = \prod_{i=1}^{|S_{c,0}|} \mathfrak{p}_i^{v_{P_i}(I)},$$

with $v_{P_i}(I) \leq v_{P_i}(c)$ for all $1 \leq i \leq |S_{c,0}|$, because I divides cO_F. Let $V_I = \begin{bmatrix} v_{P_1}(I) & \ldots & v_{P_{|S_{c,0}|}}(I) \end{bmatrix}^T$. We form $M_{S_{c,0}}$ using the precomputed matrix M_{S_c} =SValMat(S_c), which is the matrix of

the columns of M_{S_c} corresponding to the finite places in S_c. Using $M_{S_c,0}$ and the vector V_I, we can form the matrix equation below.

$$\begin{bmatrix} v_{P_1}(\epsilon_1) & v_{P_1}(\epsilon_2) & \dots & v_{P_1}(\epsilon_{|S_c|-1}) \\ v_{P_2}(\epsilon_1) & v_{P_2}(\epsilon_2) & \dots & v_{P_2}(\epsilon_{|S_c|-1}) \\ \vdots & \vdots & \ddots & \vdots \\ v_{P_{|S_{c,0}|}}(\epsilon_1) & v_{P_{|S_{c,0}|}}(\epsilon_2) & \dots & v_{P_{|S_{c,0}|}}(\epsilon_{|S_c|-1}) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{|S_c|-1} \end{bmatrix} = \begin{bmatrix} v_{P_1}(I) \\ v_{P_2}(I) \\ \vdots \\ v_{P_{|S_{c,0}|}}(I) \end{bmatrix} \tag{4.4}$$

The following lemma guarantees that I is principal if and only if the matrix equation is consistent.

Lemma 4.2. *The following are equivalent.*

(1) I is a principal ideal.

(2) equation (4.4) is consistent.

Proof. Suppose that I is a principal ideal generated by an element α. Then for any finite place $P \in \mathbb{P}_0(F)$, $v_P(\alpha) = v_P(I)$. Thus, from the construction of I, α is an S_c-unit, and $\alpha = \prod_{i=1}^{|S_c|-1} \epsilon_i^{x_{\alpha,i}}$. That means the vector $\begin{bmatrix} x_{\alpha,1} & \dots & x_{\alpha,|S_c|-1} \end{bmatrix}^T$ is a solution of (4.4). For the other direction, suppose the equation is consistent, so we can form an element using the solution, $\alpha = \prod_{i=1}^{|S_c|-1} \epsilon_i^{x_i}$. Then $\alpha O_F = I$. □

When an equation of the form (4.4) is consistent, we are likely to get a solution set of infinitely many vectors, as the number of rows is less than the number of columns. Each solution $\begin{bmatrix} x_1 & \dots & x_{|S_c|-1} \end{bmatrix}^T$ of (4.4) corresponds to $\alpha = \prod_{i=1}^{|S_c|-1} \epsilon_i^{x_i} \in O_F$ as in the proof of Lemma 4.2. That means by solving one equation of the form (4.4), we get infinitely many corresponding elements in O_F, which satisfy the following lemma.

Lemma 4.3. *Let X_I be the solution set of the matrix equation of the form (4.4) with a fixed*

vector $V_I = \begin{bmatrix} v_1 & \dots & v_{|S_{c,0}|} \end{bmatrix}^T$. *Then any two vectors*

$$\mathcal{X}_1 = \begin{bmatrix} x_{1,1} \\ \vdots \\ x_{1,|S_c|-1} \end{bmatrix} \text{ and } \mathcal{X}_2 = \begin{bmatrix} x_{2,1} \\ \vdots \\ x_{2,|S_c|-1} \end{bmatrix} \in X_I$$

correspond to two associate elements of O_F,

$$\alpha_1 = \prod_{i=1}^{|S_c|-1} \epsilon_i^{x_{1,i}} \text{ and } \alpha_2 = \prod_{i=1}^{|S_c|-1} \epsilon_i^{x_{2,i}}.$$

Proof. We know that $\mathcal{X}_1$ and $\mathcal{X}_2$ are solutions of (4.4) with fixed V_I, so

$$v_{P_j}(\alpha_1) = \sum_{i=1}^{r_{S_c}} v_{P_j}(\epsilon_i) x_{1,i} = v_j,$$
$$v_{P_j}(\alpha_2) = \sum_{i=1}^{r_{S_c}} v_{P_j}(\epsilon_i) x_{2,i} = v_j$$

for all $1 \leq j \leq |S_{c,0}|$. Thus,

$$v_{P_j}\left(\frac{\alpha_1}{\alpha_2}\right) = v_{P_j}(\alpha_1) - v_{P_j}(\alpha_2) = 0$$

for all $1 \leq j \leq |S_{c,0}|$. This means $\frac{\alpha_1}{\alpha_2}$ is a unit , thus α_1 and α_2 are associate. This means $v_P(\frac{\alpha_1}{\alpha_2}) = 0$ for any finite place $P \in \mathbb{P}_0(F)$ because α_1 and α_2 are S_c-units. This implies that $\frac{\alpha_1}{\alpha_2}$ is a unit of O_F, and hence α_1, α_2 are associate. □

Lemma 4.3 guarantees that we only need to pick one vector $\mathcal{X}$ from each solution set X_I of (4.4) with fixed V_I.

In order to compute a compact representation of a minimal solution, it is necessary to choose $\mathcal{X}_0 \in X_I$ that makes the corresponding solution α_0 minimal in the maximum additive norm. First, we choose a vector $\mathcal{X} = \begin{bmatrix} x_1 & \dots & x_{|S_c|-1} \end{bmatrix}^T \in X_I$ that is short in the Euclidean

norm and corresponds to $\alpha = \prod_{i=1}^{|S_c|-1} \epsilon_i^{x_{\alpha,i}}$. Then compute the vector $v_{\alpha,\infty}$ of the values of α at the infinite places by

$$v_{\alpha,\infty} = M_{S_c,\infty}^T \mathcal{X}. \tag{4.5}$$

From [7, Third version of Siegel's Lemma], we have

$$\|\mathcal{X}\|_\infty \leq (70\sqrt{|S_c|} d_{\max})^\xi, \tag{4.6}$$

where $d_{\max} = \left\| \left[M_{S_{c,0}}^T \;\middle|\; V \right] \right\|_\infty$, $\xi = \frac{|S_{c,0}|}{r+1}$, and r is the unit rank of O_F. That means $\|v_{\alpha,\infty}\|_\infty$ can be huge.

In order to compute a compact representation of a minimal solution α_0, we do the following. A minimal solution α_0 means α_0 is minimal in the maximum additive norm defined in (2.12). That means α_0 is minimal when the vector $v_{\alpha_0,\infty}$ of values of α_0 at the infinite places is short in the Euclidean norm. In order to find α_0, first we compute a vector v_0 in the unit lattice that is close to $v_{\alpha,\infty}$. Then we compute the vector $v_{\alpha_0,\infty}$,

$$v_{\alpha_0,\infty} = v_{\alpha,\infty} - v_0. \tag{4.7}$$

Since we know α_0 is a generator of I, we compute a compact representation $\mathbf{t}$ of α_0; that is, $\mathbf{t}$ =CompRep(I, $v_{\alpha_0,\infty}$), which is a solution of (2.7).

The algorithm and the example below describe the entire process.

The following is a small example of solving a norm equation using Algorithm 12.

Example 4.1.1. This example continues to deal with the same norm equation in Example 3.1.2 and Example 3.2.1. Let F, f, and c be as in Example 3.1.2, $F/\mathbb{F}_5(x)$ is a finite extension of degree $n = 3$ by

$$f(t) = t^3 + (4x^3 + 3x^2 + 1)t^2 + (3x^3 + 4x^2 + 4x + 2)t + 2x^3 + x,$$

and $c = x + 4$. We have 2 prime ideals that are the factors of cO_F; $\mathfrak{p}_1$ and $\mathfrak{p}_2$ that correspond

Algorithm 12 NormEquation-PrincipalIdealProblem-CR

Input: $c \in k[x] \setminus k$, the maximal order O_F of F, an S_c-unit value matrix M_{S_c}

Output: A set $\mathcal{R}$ of all non-associate solutions of (2.7) that are in O_F in compact representation.

$\mathcal{R} \longleftarrow \emptyset$
$S_{c,0} \longleftarrow \{P \in \mathbb{P}_0(F) | v_P(c) \neq 0\}$
$M_{S_c,0} \longleftarrow$ the matrix of the columns of M_{S_c} corresponding to the places in $S_{c,0}$
$M_{S_c,\infty} \longleftarrow$ the matrix of the columns of M_{S_c} corresponding to the places in $\mathbb{P}_\infty(F)$
$M_{\mathbb{P}_\infty(F)} \longleftarrow$SValMat($\mathbb{P}_\infty(F)$)
for every $I | cO_F$ such that $I = \prod_{i=1}^{|S_c|} \mathfrak{p}^{v_{P_i}(I)}$ **do**
  if $\mathrm{Norm}_{F/k(x)}(I)/c$ is in $k^\times$, **then**
    if $M_{S_c,0}^T X = V_I \left(= \begin{bmatrix} v_{P_1}(I) & \dots & v_{P_{|S_{c,0}|}}(I) \end{bmatrix}^T \right)$ is consistent, **then**
      $\mathcal{X} \longleftarrow$ a solution of $M_{S_c,0}^T X = V_I$
      $v_0 \longleftarrow$ a vector in the lattice generated by the rows of $M_{\mathbb{P}_\infty(F)}$, that is closest to $M_{S_c,\infty}^T \mathcal{X}$
      $v \longleftarrow M_{S_c,\infty}^T \mathcal{X} - v_0$
      $\mathbf{t} \longleftarrow$ CompRep(I, v)
      $\mathcal{R} \longleftarrow \mathcal{R} \cup \{\mathbf{t}\}$
    end if
  end if
end for
return $\mathcal{R}$

to the finite places P_1 and P_2. Thus, we have four ideals that divide cO_F as in Example 3.2.1,

$$I_1 = 1 \cdot O_F \qquad I_2 = \mathfrak{p}_1, \qquad I_3 = \mathfrak{p}_2, \qquad I_4 = \mathfrak{p}_1\mathfrak{p}_2.$$

We get $\mathrm{Norm}_{F/k(x)}(I_2)/c = 1 \in k^\times$. Now, we compute the S_c-unit value matrix M_{S_c} along with $M_{S_c,0}$ and $M_{S_c,\infty}$.

$$M_{S_c} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ -5 & 6 & 11 & -9 \\ 28 & -18 & 10 & -1 \end{bmatrix}, \qquad M_{S_c,0} = \begin{bmatrix} 1 & 1 \\ -5 & 6 \\ 28 & -18 \end{bmatrix}, \qquad M_{S_c,\infty} = \begin{bmatrix} -1 & -1 \\ 11 & -9 \\ 10 & -1 \end{bmatrix},$$

Then we form a matrix equation $M_{S_c,0}^T X = V_{I_2}$,

$$\begin{bmatrix} 1 & -5 & 28 \\ 1 & 6 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

which has the solution set

$$X_{I_2} = \left\{ \begin{bmatrix} 114 \\ -67 \\ -16 \end{bmatrix} + a \begin{bmatrix} 78 \\ -46 \\ -11 \end{bmatrix} \;\middle|\; a \in \mathbb{Z} \quad \right\}. \tag{4.8}$$

We simply pick a solution from the above set, $\mathcal{X} = \begin{bmatrix} 114 & -67 & -16 \end{bmatrix}^T$. Then

$$v_{\alpha,\infty} = M_{S_c,\infty}^T \mathcal{X} = \begin{bmatrix} -1 & 11 & 10 \\ -1 & -9 & -1 \end{bmatrix} \begin{bmatrix} 114 \\ -67 \\ -16 \end{bmatrix} = \begin{bmatrix} -1011 \\ 505 \end{bmatrix}.$$

Now we want to find a vector v_0 in the unit lattice that is close to $\begin{bmatrix} -1011 & 505 \end{bmatrix}^T$. The unit lattice is generated by the rows of $M_{\mathbb{P}_\infty(F)}$ =SValMat($\mathbb{P}_\infty(F)$),

$$M_{\mathbb{P}_\infty(F)} = \begin{bmatrix} 694 & -347 \end{bmatrix}.$$

In fact, $v_0 = \begin{bmatrix} -694 & 347 \end{bmatrix}^T$ is a vector in the unit lattice closest to $\begin{bmatrix} -1011 & 505 \end{bmatrix}^T$. Thus, we compute

$$v_{\alpha_0,\infty} = M^T_{S_{c,\infty}}\mathcal{X} - v_0 = \begin{bmatrix} -1011 \\ 505 \end{bmatrix} - \begin{bmatrix} -694 \\ 347 \end{bmatrix} = \begin{bmatrix} -317 \\ 158 \end{bmatrix},$$

and compute a compact representation $\mathbf{t}$=CompRep(I_2, $\begin{bmatrix} -317 & 158 \end{bmatrix}$), that is the same as in Example 3.2.1. Thus, $\{\mathbf{t}\}$ is the set of non-associate solution of equation (2.7) with $c = x + 4$. The entire process took 0.180 CPU seconds on an Intel Xeon CPU E7-8891 v4 with 80 64-bit cores at 2.80GHz.

While the same machine was unable to solve the norm equation using Algorithm 10 for 4 days, and took 114.830 CPU seconds using Algorithm 11, it only took 0.180 CPU seconds solving it using Algorithm 12. In the following section, we provide detailed complexity analysis on Algorithm 12 and compare it to the complexity of Algorithm 10 and Algorithm 11.

Considering that the solution α of (2.7) is an S_c-unit as written in (4.3), a part of Algorithm 12 can be performed differently; computing a compact representation of α. We briefly introduce the variation before analyzing complexity of Algorithm 12.

A variation of Algorithm 12

In this subsection, we describe a variant of Algorithm 12 for solving (2.20). This variant follows the framework of Algorithm 12, which enumerates ideals dividing cO_F, computes their norms, and computes compact representations of the solutions of (2.20). The differences

between this variant and Algorithm 12 are that (1) this variant computes compact representations of a set of fundamental S-units, (2) uses them to compute compact representations of the solutions of (2.20) using Algorithm 5 and Algorithm 6, and (3) does not require to compute close vectors because this variant do not find minimal solutions of (2.7). Recall that one of our goals was to represent solutions of (2.7) in smaller sizes, and the cost of computing close vectors is not the dominating factor of the overall complexity of Algorithm 12. Since this variant is largely the same as Algorithm 12, but does not produce minimal solutions, we only describe how some parts of Algorithm 12 can be performed in different ways and provide an example, but do not present entire algorithms with the different parts incorporated. We provide the complexity of the parts at the end of Section 4.1.2.

We first compute compact representations of a system of fundamental S_c-units $\{\epsilon_i | 1 \leq i \leq |S_c| - 1\}$ using the matrix M_{S_c} and Algorithm 4, and denote the compact representation of each ϵ_i by $\mathbf{t}_i$ for $1 \leq i \leq |S_c| - 1$. Then we check the norm of I, and we solve the same matrix equations $M_{S_c,0}^T X = V$ of the form (4.4) and get the solution sets X_V. By Lemma 4.3, we only pick one vector $\mathcal{X} = \begin{bmatrix} x_1 & \dots & x_{r_{S_c}} \end{bmatrix}^T$ in each X_V to form S_c-units $\alpha = \prod_{i=1}^{|S_c|-1} \epsilon_i^{x_i}$. When picking $\mathcal{X}$, we preferably choose a short vector in X_V. We mean short in the Euclidean norm here so that we compute a compact representation of α more efficiently. From the form in (4.3), we compute powers of the compact representations $\prod_{i=1}^{|S_c|-1} \epsilon_i^{x_i}$ using Algorithm 6, and multiply them using Algorithm 5. Then we obtain a compact representation of α, a solution of (2.7).

The example below shows how to compute compact representations using Algorithm 6 and Algorithm 5.

Example 4.1.2. Let f, F, c be as in Example 4.1.1. With the same ideals I_1, I_2, I_3, I_4, and the same matrix equation $M_{S_c,0}^T X = V_{I_2}$, and the solution set X_{I_2} as in Example 4.1.1, we pick a short vector $\mathcal{X}_0 = \begin{bmatrix} 36 & -21 & -5 \end{bmatrix}$. With the vector, we know that $\alpha = \epsilon_1^{36} \epsilon_2^{-21} \epsilon_3^{-5}$ is a solution of the norm equation. In order to compute a compact representation of α using Algorithm 5 (MultCR) and Algorithm 6 (PowCR), we first compute compact representations

of a system of fundamental S-units.

Let $\{\epsilon_1, \epsilon_2, \epsilon_3\}$ be a system of fundamental S_c-units. Then due to construction, for $1 \leq i \leq 3$, we have

$$\epsilon_i O_F = \mathfrak{p}_1^{(M_{S_c,0})_{i,1}} \mathfrak{p}_2^{(M_{S_c,0})_{i,2}},$$

and $v_{P_{1,\infty}}(\epsilon_i) = (M_{S_c,\infty})_{i,1}$ and $v_{P_{2,\infty}}(\epsilon_i) = (M_{S_c,\infty})_{i,2}$. Thus, CompRep($\mathfrak{p}_1^{(M_{S_c,0})_{i,1}} \mathfrak{p}_2^{(M_{S_c,0})_{i,2}}$, $(M_{S_c,\infty})_i$) is a compact representation of ϵ_i and we denote it by $\mathbf{t}_i$, so $\mathbf{t}_1$=CompRep($\mathfrak{p}_1\mathfrak{p}_2$, $\begin{bmatrix} -1 & -1 \end{bmatrix}$) of ϵ_1, $\mathbf{t}_2$=CompRep($\mathfrak{p}_1^{-5}\mathfrak{p}_2^{6}$, $\begin{bmatrix} 11 & -9 \end{bmatrix}$) of ϵ_2, and $\mathbf{t}_3$=CompRep($\mathfrak{p}_1^{28}\mathfrak{p}_2^{-18}$, $\begin{bmatrix} 10 & -1 \end{bmatrix}$) of ϵ_3 as follows.

$\epsilon_1 = \mu_1 \prod_{i=1}^{4} \left(\frac{1}{\beta_{1,i}}\right)^{2^{l-i}}$, where

$$\begin{aligned}
\mu_1 &= (2x+3)y^2 + (4x+1)y + x + 4, \\
\beta_{1,1} &= \beta_{1,2} = 1, \\
\beta_{1,3} &= y^2 + 2y + 3, \\
\beta_{1,4} &= \frac{4x+4}{x^3+3x^2+4x+4} y^2 + \frac{x^3+x^2+1}{x^2+x+2} y + \frac{x^3+x^2+3}{x^3+3x^2+4x+4}.
\end{aligned}$$

$\epsilon_2 = \mu_2 \prod_{i=1}^{5} \left(\frac{1}{\beta_{2,i}}\right)^{2^{l-i}}$, where

$$\begin{aligned}
\mu_2 =& \frac{4x^8+4x^7+x^5+2x^4+3x^3+2}{x^3+2x^2+3x+4}y^2 + \frac{3x^8+x^7+x^6+3x^4+4x^2+x+1}{x^3+2x^2+3x+4}y \\
&+ \frac{2x^8+3x^7+2x^5+2x^3+3}{x^3+2x^2+3x+4}, \\
\beta_{2,1} =& 2y^2+4y+1, \\
\beta_{2,2} =& \frac{x+1}{x^3+3x^2+4x+4}y^2 + \frac{4x^3+4x^2+4}{x^2+x+2}y + \frac{4x^3+4x^2+2}{x^3+3x^2+4x+4}, \\
\beta_{2,3} =& \frac{4x+2}{x^4+2x+3}y^2 + \frac{x^4+x^2+4}{x^4+2x+3}y + \frac{x^4+3x^3+3x+1}{x^4+2x+3}, \\
\beta_{2,4} =& \frac{3x^4+3x^2+4x+4}{x^8+4x^7+x^6+4x^5+3x^4+4x^3+2x^2+1}y^2 + \frac{2x^7+4x^6+2x^5+3x^3+2x^2+2x+1}{x^8+4x^7+x^6+4x^5+3x^4+4x^3+2x^2+1}y \\
&+ \frac{2x^7+2x^5+x^4+3x^3+3x+2}{x^8+4x^7+x^6+4x^5+3x^4+4x^3+2x^2+1}, \\
\beta_{2,5} =& \frac{3x^6+4x^5+2x^4+3x^2+3x+4}{x^8+2x^7+4x^4+x^3+2x^2+2x+4}y^2 + \frac{2x^7+4x^5+4x^4+4x^2+2}{x^6+2x^5+2x^4+4x^3+3x^2+4x+3}y \\
&+ \frac{4x^7+3x^6+2x^5+x^4+2x^2+1}{x^6+2x^5+2x^4+4x^3+3x^2+4x+3}.
\end{aligned}$$

$\epsilon_3 = \mu_3 \prod_{i=1}^{4} \left(\frac{1}{\beta_{3,i}}\right)^{2^{l-i}}$, where

$$\begin{aligned}
\mu_3 =& (4x^{15}+2x^{12}+3x^{11}+2x^9+2x^8+4x^7+4x^6+x^5+x^2+x+3)y^2 \\
&+ (x^{18}+2x^{17}+x^{15}+2x^{14}+2x^{12}+4x^{10}+3x^9+3x^8+3x^7+3x^6+x^5+4x^4+2x^3+3x^2+4x+4)y \\
&+ \frac{3x^{18}+4x^{17}+2x^{16}+4x^{15}+x^{14}+x^{13}+3x^{12}+x^{11}+3x^9+3x^8+4x^6+4x^4+4x^3+2x^2+x+1}{x^{18}+2x^{17}+3x^{16}+4x^{15}+2x^{13}+4x^{12}+x^{11}+3x^{10}+3x^8+x^7+4x^6+2x^5+4x^3+3x^2+2x+1}, \\
\beta_{3,1} =& \beta_{3,2} \\
\beta_{3,3} =& y^2+2y+3, \\
\beta_{3,4} =& \frac{4x+4}{x^3+3x^2+4x+4}y^2 + \frac{x^3+x^2+1}{x^2+x+2}y + \frac{x^3+x^2+3}{x^3+3x^2+4x+4}.
\end{aligned}$$

MultCR(MultCR(PowCR($\mathbf{t}_1$, 36), PowCR($\mathbf{t}_2$, -21)), PowCR($\mathbf{t}_3$, -5)) is a compact representation of α.

Using this variation can help avoid computing a close vector, but the solution may not be

minimal, and the compact representation obtained from Algorithm 5 and Algorithm 6 can be large. The cost of computing such a compact representation and the size of the solutions are provided at the end of Section 4.1.2.

4.1.2 Complexity analysis

In this subsection, we investigate the expected running time of Algorithm 12 described in Section 4.1.1, under the assumptions that F has at least one infinite place of degree 1. The assumption is required to compute compact representations. We provide the complexity of Algorithm 12 in the number of bit operations. Using the algorithms in [23], an operation in k takes $O((\log q)^{1+\varepsilon}) = O(q^{\varepsilon})$ bit operations.

Throughout this section, we let $\Lambda \in \mathbb{Z}^r$ be the unit lattice of O_F, which is generated by an LLL-reduced basis $\{(v_{P_1}(\varepsilon_j), \ldots, v_{P_r}(\varepsilon_j)) \in \mathbb{Z}^r | 1 \leq j \leq r\}$, where $\{\varepsilon_j | 1 \leq j \leq r\}$ is a set of fundamental units and $\{P_1, \ldots, P_{r+1}\}$ is the set of the infinite places of F with $\deg P_{r+1} = 1$. We also let Λ_{S_c} be the S_c-unit lattice with an LLL-reduced basis $\{b_1, \ldots, b_{|S_c|-1}\}$, and $\{\epsilon_i | 1 \leq i \leq |S_c| - 1\}$ is a set of fundamental S_c-units such that $b_i = (v_{P_1}(\epsilon_i), \ldots, v_{P_r}(\epsilon_i))$. We define $d_{\max}$, d_ϵ, and d_N as follows.

$$
\begin{aligned}
d_{\max} &= \left\| \left[M_{S_{c,0}}^T \;\middle|\; V \right] \right\|_\infty, \\
d_\epsilon &= \max \left\{ \mathbf{h}(\epsilon_1), \;\; \ldots, \;\; \mathbf{h}(\epsilon_{|S_c|-1}), 2 \left(\left\lceil \frac{2g-1}{n} \right\rceil + 1 \right) \right\}, \\
d_N &= \max_{1 \leq i \leq r} \{ \mathbf{h}(\mathrm{Norm}_{F/k(x)}(\epsilon_i)) \}.
\end{aligned}
$$

Note that all three variables defined above are bounded as below.

$$
\begin{aligned}
d_{\max}, d_\epsilon, d_N &\leq \max \left\{ \left\| \left[M_{S_c}^T \right] \right\|_\infty, \;\; \max_{P \in S_{c,0}} v_P(c), \;\; 2 \left\lceil \frac{2g-1}{n} \right\rceil + 2 \right\} \\
&\leq \max \left\{ 2^{(|S_c|-1)(|S_c|-2)/4} R'_S, \;\; \max_{P \in S_{c,0}} v_P(c) \right\}, \qquad (4.9)
\end{aligned}
$$

where R'_S is as defined in (2.20). The last inequality is by Proposition 2.29. We let

$$d_{S_c} := \max\left\{2^{(|S_c|-1)(|S_c|-2)/4}R'_S,\ \max_{P\in S_{c,0}} v_P(c)\right\}. \tag{4.10}$$

Note that d_{S_c} only depends on c and the function field F.

Next, we analyze the cost of Algorithm 12.

The number of I that divide cO_F is the same as in Lemma 3.8. For each I that divides cO_F, we have five components to compute: the HNF representation of I, $\mathrm{Norm}_{F/k(x)}(I)$, solving the matrix equation (4.4), searching for a vector v_0 in the unit lattice that is close to $M^T_{S_c,\infty}\mathcal{X}$ where $\mathcal{X}$ is a solution of (4.4), and computing compact representations for the solutions.

The HNF representation of I is as defined in Definition 2.16. Each ideal I has the form $I = \prod_{i=1}^{|S_{c,0}|}\mathfrak{p}_i^{v_i}$ as in (4.2), where $\mathfrak{p}_i$ are the prime ideals of F corresponding to the places P_i in $S_{c,0}$, for $1 \le i \le |S_{c,0}|$. We enumerate all such I. The HNF representation of I can be obtained by computing the HNF representation of the product of two ideals. From [44, Theorem 5.2.2.], it costs

$$O\left(n^7g^2q^\varepsilon\right) \tag{4.11}$$

bit operations, when n and g go to infinity.

The cost of computing the norm of I is given in Lemma 2.37. Since c is in $k[x]$, each I is an integral ideal. That means the denominator of I is 1. Thus, it takes

$$O\left(\left(n^{2+\varepsilon}\deg c + (n\deg c)^{1+\varepsilon}\right)q^\varepsilon\right) \tag{4.12}$$

bit operations to compute $\mathrm{Norm}(I)$.

Now we consider the cost of solving a matrix equations of the form of (4.4). With the same notation from the previous section, the matrix equation is $M^T_{S_c,0}X = V$, where $M^T_{S_c,0}$

is an $|S_{c,0}| \times |S_c| - 1$ matrix and $V \in \mathbb{Z}^{|S_{c,0}|}$. From the construction, we know

$$\left\|M_{S_c,0}^T\right\|_\infty = \max_{\substack{P\in\mathbb{P}_\infty(F)\\ 1\le i\le |S_c|-1}} v_P(\epsilon_i) \quad \text{and} \quad \|V\|_\infty \le \max_{P\in\mathbb{P}_0(F)} v_P(c).$$

Then by [9, Theorem 22], the cost of solving the integer matrix $M_{S_c,0}^T X = V$ is

$$O(|S_c|^{\omega+\epsilon} d_{\max}^{\epsilon}) \tag{4.13}$$

bit operations, where ω is the matrix multiplication exponent.

The expected running time of searching a vector v_0 in the unit lattice Λ closest to $M_{S_c,\infty}^T$ by the algorithm given in [8] is

$$O(2^{0.3774r}). \tag{4.14}$$

Moreover, we have

$$\left\|M_{S_c,\infty}^T \mathcal{X} - v_0\right\|_\infty \le \max_{1\le i\le r} \left\{ \sum_{j=1}^{r} \frac{1}{2} |v_{P_i}(\varepsilon_j)| \right\}.$$

The cost of computing a compact representation $\mathbf{t}$ for a solution corresponding to the solution $\mathcal{X}$ such that $M_{S_c,0}\mathcal{X} = V$ is the cost of Algorithm 4 with inputs $A = \prod_{j=1}^{|S_{c,0}|} \mathfrak{p}_j^{(V)_j}$, $M_{S_c,\infty}^T \mathcal{X} - v_0$, where $\mathfrak{p}_j$ is the prime ideal corresponding to each place $P_j \in S_{c,0}$. We know that

$$\mathbf{h}(\mathrm{div}(A)) = \sum_{i=1}^{|S_{c,0}|} |(V)_j| \deg \mathfrak{p}_j \le \sum_{i=1}^{|S_{c,0}|} |v_{P_j}(c)| \deg \mathfrak{p}_j = \mathbf{h}(\mathrm{div}(cO_F)). \tag{4.15}$$

Since $\mathrm{Norm}(A) = \zeta c$ for some $\zeta \in k^\times$, $\mathbf{h}(\mathrm{Norm}(A)) = \deg c$. Thus, computing a compact representation of a solution of (2.7) takes

$$O\left(gq^{\varepsilon}\left(\mathsf{RR}\left(n\deg c + g\right) + \log\left(d_{S_c} + n + g\right)\mathsf{RR}\left(n^2 + ng\right)\right)\right) \tag{4.16}$$

bit operations as in Lemma 2.42, where $\mathsf{RR}(\cdot)$ is the cost of computing a Riemann-Roch

space defined in (2.33).

When $\deg(c)$ is fixed and n is very large, the cost of the algorithm is dominated by computing M_{S_c} and v_0, because the number of bit operations needed to compute all other components is polynomial in n. The following theorem provides the asymptotic cost of Algorithm 12 for fixed $\deg(c)$ and large n.

Theorem 4.4. *With d_{S_c} defined as in (4.10), Algorithm 12 solves a norm equation (2.7) in*

$$O\Bigg(\left(\prod_{P\in S_{c,0}}(v_P(c)+1)\right)\left(|S_c|^{\omega+\varepsilon}d_{s_c}^{\varepsilon}+2^{0.3774r}\right.$$
$$\left.+q^{\varepsilon}\left(n^7g^2+n^{2+\varepsilon}(\deg c)^{1+\varepsilon}+g\mathsf{RR}\left(n\deg c+g\right)+g\log(d_{S_c}+n+g)\mathsf{RR}\left(n^2+ng\right)\right)\right)\Bigg)$$

bit operations, when n, g, $\deg c$, and $q\to\infty$.

Proof. In this algorithm, we compute M_{S_c} and $M_{\mathbb{P}_\infty(F)}$, then run the iteration in Step 6-16. Each iteration consists of enumerating I, computing the norm of I, solving the matrix equation, computing v_0, and computing a compact representation, which takes

$$O(\left(n^7g^2+n^{2+\varepsilon}\deg c+(n\deg c)^{1+\varepsilon}\right)q^{\varepsilon}+|S_c|^{\omega+\varepsilon}d_{\max}^{\varepsilon}+2^{0.3774r}$$
$$+gq^{\varepsilon}(\mathsf{RR}\left(n\deg c+g\right)+\log(d_{S_c}+n+g)\mathsf{RR}\left(n^2+ng\right)))$$

bit operations. We know $d_{\max}\le d_{S_c}$. Since $n^{1+\varepsilon}$ is clearly less than $n^{\omega+\varepsilon}$, the second and third terms can be simplified to $n^{2+\varepsilon}\deg c^{1+\varepsilon}$. The number of iterations is the number of ideals I that divides cO_F, which is $\prod_{P\in S_{c,0}}(v_P(c)+1)$ from Lemma 3.8. Combining all the costs, the result is obtained as stated. □

Each solution obtained from Algorithm 12 has the same size as those obtained from Algorithm 11. Lemma 3.10 gives the bounds on their sizes.

We discuss the worst case complexity of Algorithm 12 when each of n, $\deg c$ g and q goes to infinity and the rest are fixed.

In order to simplify the complexity in Theorem 4.4, we bound the number of iterations, $\prod_{P\in S_{c,0}}(v_P(c)+1)$, and R'_S. From the Remarks below Lemma 3.8, when cO_F is irreducible and the place P above cO_F is unramified, the number of I is 1 and $v_P(c)=1$. Thus, we have the same bound as in (3.25),

$$2 \leq \prod_{P\in S_{c,0}} (v_P(c)+1) \leq 2^{n\deg c}. \tag{4.17}$$

From Lemma 2.28 and (2.17), we have the following bound on R'_S,

$$R'_S \leq h_{O_F} R_F \leq (\sqrt{q}+1)^{2g}. \tag{4.18}$$

When n goes to infinity and the other variables are fixed, the asymptotic complexity is as follows. From the bound in (4.17), we have $\prod_{P\in S_{c,0}}(v_P(c)+1) = O(2^{n\deg c})$. Moreover, $d_{S_c} = O(2^{n^2}R'_S) = O(2^{n^2})$. Using the cost of computing Riemann-Roch space in (2.33), $\mathsf{RR}(n^2+ng) = O(n^9)$. The asymptotic complexity of Algorithm 12 is

$$O\left(2^{n(0.3774+\deg c)}\right) = 2^{n(\deg c + o(1))}, \tag{4.19}$$

which is exponential in n, because the number of ideals above cO_F is exponential in n. Note that when c and cO_F are irreducible, we have $\prod_{P\in S_{c,0}}(v_P(c)+1)=2$ as in (4.17), the asymptotic complexity is

$$O\left(2^{0.3774n}\right),$$

which is still exponential in n, but smaller than (4.19) by $2^{n\deg c}$ factor. Compared to the exhaustive search algorithms, Algorithm 10 and Algorithm 11, which were doubly exponential and superexponential in n, respectively, Algorithm 12 is much faster for large n. This is mainly because the number of ideals to be tested in Algorithm 12 is $O(2^{n\deg c})$, and this number is smaller than the number of elements in the search spaces of Algorithm 10 by a doubly exponential factor and of Algorithm 11 by a superexponential factor, as Algorithm 10

and Algorithm 11 have $O(2^{n^2 R_F 2^{n^2}/4} q^{\varepsilon})$ and $O(2^{n^3/4+n\deg c+n^{1+\varepsilon}} R_F^n)$ elements in their search spaces. Due to the sizes of the search spaces, the overall complexity of Algorithm 12 is also smaller than Algorithm 10 by a doubly exponential factor, and is smaller than Algorithm 11 by a superexponential factor.

When g goes to infinity and the other variables are fixed, $\mathsf{RR}(n^2+ng) = O(g^{2+\varepsilon})$, and $\log(d_{S_c}+n+g) = O(g^{\varepsilon} R_S'^{\varepsilon}) = O(g^{1+\varepsilon})$, from (4.18). Thus, the asymptotic complexity of Algorithm 12 is

$$O\left(g^{4+\varepsilon}\right),$$

which is polynomial in g. This is smaller than the precomputation of computing M_{S_c}, which is subexponential in g by Lemma 2.38. It is doubly exponentially and exponentially smaller than Algorithm 10 and Algorithm 11, respectively, as they are doubly exponential and exponential in g. This is mainly because of the number of elements or ideals to be tested in each algorithm. The number of ideals to be tested in Algorithm 12 does not depend on g, therefore Algorithm 12 has no factor related to the number of ideals. On the other hand, Algorithm 10 and Algorithm 11 have $O(2^{n^2 q^g 2^{n^2}/4} q^{\varepsilon})$ and $O(R_F^n) = O(q^{gn}) = O(2^{gnq^{\varepsilon}})$ elements in their search spaces, respectively. Similar to Algorithm 11, using compact representations improves the computational cost of Algorithm 12. Algorithm 12 computes a compact representation and its norm in $O(g^{4+\varepsilon})$ bit operations. This is much less than $O(q^{g+\varepsilon})$, which is the number of bit operations Algorithm 10 needs to compute the norm of an element in its search space.

When $\deg c$ goes to infinity, and the other variables are fixed, we have $\prod_{P\in S_{c,0}}(v_P(c)+1) = O(2^{n\deg c})$, $d_{S_c} = O(2^{(\deg c)^2} R_S') = O(2^{(\deg c)^2})$. Using the cost of computing Riemann-Roch spaces in (2.33), $\mathsf{RR}(n\deg c+g) = O((\deg c)^2)$. The asymptotic complexity of Algorithm 12 is

$$O\left(2^{n\deg c}(\deg c)^{\omega+2+\varepsilon}\right),$$

which is exponential in $\deg c$. The exponential factor $2^{n\deg c}$ is from the number of I dividing

cO_F, and the polynomial factor $(\deg c)^{\omega+2+\varepsilon}$ is from solving the matrix equations (4.4). Similar to the analysis for large n, when c and cO_F are irreducible, we have $\prod_{P\in S_{c,0}}(v_P(c)+1)=2$. In addition, in that case, $|S_c|$ and d_{S_c} are constant. Thus, the asymptotic complexity of Algorithm 12 when c and cO_F are irreducible is

$$O((\deg c)^2),$$

which is dominated by the cost of computing compact representations. The complexity is smaller than Algorithm 10 by $2^{q^\varepsilon \deg c}(\deg c)^{\omega-2}$ factor, and Algorithm 11 by $(\deg c)^{\omega-2}$ factor, as they are $O(2^{q^\varepsilon \deg c}(\deg c)^\omega)$ and $O((\deg c)^\omega)$, respectively. Note that $(\deg c)^{\omega-2} \approx (\deg c)^{0.37}$, using the up-to-date matrix multiplication exponent in [29, Proposition 3.3].

When q goes to infinity and the other variables are fixed, the asymptotic complexity is

$$O\left(q^\varepsilon\right).$$

This is smaller than the precomputation of M_{S_c}, because the cost of computing M_{S_c} is polynomial in q by Lemma 2.38. Compared to the costs of the other algorithms, the asymptotic complexity of Algorithm 12 is smaller than Algorithm 10 by a superexponential factor, and Algorithm 11 by a polynomial factor.The asymptotic complexity of Algorithm 12 is small because the number of ideals to be tested in the algorithm does not depend on q. That means the asymptotic complexity of Algorithm 12 when q goes to infinity has no factor related to the number of ideals, while Algorithm 10 and Algorithm 11 have $O(q^{n^2R_F2^{n^2/4}})$ and $O(q^{gn+\varepsilon})$, respectively, from the number of elements in their search spaces.

Table 4.1 summarizes the asymptotic complexities of Algorithm 10, Algorithm 11, and Algorithm 12.

In conclusion, the worst case complexity of Algorithm 12 is asymptotically smaller in n, g, and q, than the complexity of the exhaustive search algorithms in Chapter 3. The reason why Algorithm 12 is faster is because the number of ideals tested in Algorithm 12 is far

Algorithm 10	Algorithm 11	Algorithm 12
$n \to \infty$		
$2^{O\left(n^2 2^{n^2/4}\right)}$	$2^{n^3(1+o(1))}$	$2^{n(\deg c+o(1))}$
$n \to \infty$, c, cO_F irreducible (best)		
$2^{O\left(n^2 2^{n^2/4}\right)}$	$2^{n^3(1+o(1))}$	$O(2^{0.3774n})$
$g \to \infty$		
$2^{O(q^g)}$	$2^{O(g)}$	$O(g^{4+\varepsilon})$
$\deg c \to \infty$		
$O\left(2^{q^\varepsilon \deg c}(\deg c)^\omega\right)$	$O(2^{n \deg c}(\deg c)^\omega)$	$O(2^{n \deg c}(\deg c)^{\omega+2+\varepsilon})$
$\deg c \to \infty$, c irreducible (best)		
$O\left(2^{q^\varepsilon \deg c}(\deg c)^\omega\right)$	$O((\deg c)^\omega)$	$O((\deg c)^2)$
$q \to \infty$		
$q^{O(q^g)}$	$O(q^{gn+\varepsilon})$	$O(q^\varepsilon)$

Table 4.1: Summary of asymptotic complexity of Algorithm 10, Algorithm 11, and Algorithm 12

less than the elements in the search spaces o the exhaustive search algorithms. In addition, Algorithm 11 is even faster when c and cO_F have fewer factors.

Complexity analysis of the variant of Algorithm 12

We briefly provide the cost of the variant described at the end of Section 4.1.1, using Algorithm 6 and Algorithm 5 to compute compact representations of solutions. As mentioned before, this variant can be useful if compact representations of a set of fundamental S-units are already computed. This variant does not compute close vectors, thus, the solutions from this variants are not necessarily minimal. Note that one of our goals was to represent solutions of (2.7) in smaller sizes. Thus, we do not consider this variant over Algorithm 12. We only provide the costs of the parts that are different from Algorithm 12, and the size of the solutions computed from this variant, but do not provide the total number of bits required for this variant.

Let $\{\epsilon_1, \ldots, \epsilon_{|S_c|-1}\}$ be a system of fundamental S_c-units. The cost of computing compact representation $\mathbf{t}_i$ for each ϵ_i is the cost of Algorithm 4 with inputs $A = \prod_{j=1}^{|S_{c,0}|} \mathfrak{p}_j^{(M_{S_c,0})_{i,j}}$, $v = (M_{S_c,\infty})_i$, where $\mathfrak{p}_j$ is the prime ideal corresponding to the place $P_j \in S_{c,0}$. The cost of computing each $\mathbf{t}_i$ is

$$O\left(gq^{\varepsilon}\left(\mathsf{RR}\left(n\sum_{j=1}^{|S_{c,0}|}|(M_{S_{c,0}})_{i,j}|\deg P_j + g\right) + \mathsf{RR}\left(n^2+ng\right)\log\left\|(M_{S_{c,\infty}})_i\right\|_{\infty}\right)\right)$$

bit operations, as in Lemma 2.42.

Next, we analyze the cost of computing $\alpha \in F$ with a short solution $\mathcal{X}$ of the matrix equation (4.4). From [7, Third version of Siegel's Lemma], we have a bound on $\mathcal{X}$ in (4.6). We compute $\alpha = \prod_{i=1}^{|S_c|-1} \epsilon_i^{\mathcal{X}_i}$ for each consistent matrix equation. Computing α takes $O(n^3 q^{\varepsilon} |S_c|^{2+\epsilon}(\|\mathcal{X}\|_{\infty} d_{\epsilon})^{1+\epsilon})$ bit operations by Lemma 2.32 and Lemma 2.34. In addition, $\mathbf{h}(\alpha)$ is in $O(|S_c|\,\|\mathcal{X}\|_{\infty} d_{\epsilon})$.

Now, we consider the cost of computing a compact representation $\mathbf{t}$ of the corresponding

solution. Computing $\mathbf{t}$ involves Algorithm 5 and Algorithm 6, namely MultCR and PowCR, up to $|S_c| - 1$ times each. The costs of MultCR and PowCR are as in Lemma 2.46 and Lemma 2.47. Let $l_{max} = \max_{1 \le i \le |S_c|-1}\{l_{\mathbf{t}_i}\}$. Then computing $\mathbf{t}$ takes

$$O\left(|S_c| q^{\varepsilon}\left(n^{4+\varepsilon}|S_c|\,\|\mathcal{X}\|_{\infty}^{1+\varepsilon}\, d_N^{1+\varepsilon} + l_{\max}\,\|\mathcal{X}\|_{\infty}^{1+\varepsilon}\left(n^5 + n^4 g + n^5 C_f\right)^{1+\varepsilon}\right)\right)$$

bit operations, where $d_N = \max_{1 \le i \le |S_c|-1} \mathbf{h}(\mathrm{Norm}_{F/k(x)}(\epsilon_i))$.

Lastly, combining the bounds in Lemmas 2.46 and 2.47, the S-unit variant using Algorithms 5 and 6 computes compact representations of solutions of Equation (2.7) satisfying the following lemma.

Lemma 4.5. *Let $\xi = \frac{|S_{c,0}|}{r+1}$ and $d_N = \max_{1 \le i \le |S_c|-1} \mathbf{h}(Norm_{F/k(x)}(\epsilon_i))$. If (2.7) has a solution, a compact representation $\mathbf{t} = (\mu, \beta_1, \ldots, \beta_l)$ of a solution obtained from this variant satisfies the following.*

$$\mathbf{h}(\mu) = O(|S_c|\,\|\mathcal{X}\|_{\infty}\,(nd_N + g + n^2 C_f)) = O(70^{\xi}|S_c|^{\xi/2+1} d_{\epsilon}^{\xi}(nd_N + g + n^2 C_f))$$
$$\mathbf{h}(\beta_i) = O(|S_c|\,\|\mathcal{X}\|_{\infty}\,(n^2 + ng + n^2 C_f)) = O(70^{\xi}|S_c|^{\xi/2+1} d_{\epsilon}^{\xi}(n^2 + ng + n^2 C_f)),$$

where $l \le \max_{1 \le i \le |S_c|-1} l_{\mathbf{t}_i}$, and $\|\mathcal{X}\|_{\infty}$ is bounded as in (4.6).

As mentioned before, using this variant can help avoid computing a close vector, which can dominate the overall asymptotic complexity when n goes to infinity. As seen in Lemma 4.5, solutions produced by this variant tend to be larger than those by Algorithm 12, given in Lemma 3.10, as the heights in Lemma 4.5 have an extra factor of $70^{\xi}|S_c|^{\xi/2+1} d_{\epsilon}^{\xi}$.

Chapter 5

Empirical analysis

In this chapter, we present our numerical experiments on the algorithms described in Chapter 3 and Chapter 4: the Gaál-Pohst method (Algorithm 10), our first new algorithm inspired by Gaál-Pohst using compact representations (Algorithm 11), and our second new algorithm using the principal ideal generator approach and compact representations (Algorithm 12). The algorithms are all implemented and were tested in the high-level computer algebra system Magma [10]. The Magma implementations for all the algorithms and testing code are available at https://github.com/s-leem/FF_NormEq_CR. All experiments were performed on an Intel Xeon CPU E7-8891 v4 with 80 64-bit cores at 2.80GHz.

The implemented algorithms were thoroughly tested by black-box and white-box testing to ensure the correctness of their implementations. For the black-box testing, we solved numerous norm equations with randomly generated c and F with $\deg c$, n, q and g varied. In order to verify that Algorithm 11, Algorithm 12, and Algorithm 10 were implemented correctly, we first created an extra algorithm using the existing principal ideal tests available in Magma with the framework of Algorithm 12. Then for a number of norm equations with varied $\deg c$, n, q and g, we ensured that the number of solutions from all four algorithms matches and solutions in a solution set from each algorithm are non-associate, and verified that the elements in the solution sets in fact have norm c, up to non-zero constants. White-

box testing tests every path of each algorithm. For each algorithm, we carefully divided it into cases, ran the algorithm with randomly generated inputs for each case, and compared the solution sets as we did for black-box testing.

Timing tests were conducted for all three algorithms to compare their running times and to observe how each algorithm scales as various parameters increase. In order to compare the performance of Algorithm 10, Algorithm 11, and Algorithm 12, we timed them under different settings; varying one of n, g, q, $\deg c$, and the number of factors of c, while the other parameters were fixed. Since Algorithm 10 was much slower than the others, we conducted two or three tests for each varied parameter, except for q; one with minimal fixed parameters for all three algorithms and the other(s) with relatively bigger fixed parameters for the faster algorithm(s). For q, we conducted only one test, because Algorithm 10 was very slow even with the minimal fixed parameters. We used the same fixed parameters and timed Algorithm 11 and Algorithm 12 up to 14-bit prime q, instead of using different fixed parameters. All timing tests were performed with fields F that satisfy the following: the base field k is a finite field $\mathbb{F}_q$ where q is a prime and the greatest common divisor of q and n is 1, and at least one infinite place of F has degree 1. Moreover, we ensured that the ideal class number h_{O_F} was one unless specified otherwise. The reason why we used F with $h_{O_F} = 1$ is because h_{O_F} affects the bounds of the regulator R_F in (2.17), therefore, affects the running time of the algorithms. The timing results are presented as the average number of CPU seconds of randomly generated examples of the indicated n, g, q, $\deg c$, and the number of factors of c. Note that tests were forcefully terminated when the CPU time required for each algorithm and its precomputation were longer than 1 day. In addition, although S-unit valuation matrices are precomputed for all of Algorithm 10, Algorithm 11, and Algorithm 12, this computation is expected to take longer than Algorithm 12 in some cases from the complexity analysis. Thus, we measured the CPU time it took to precompute the matrices and compared it to the timing results of the algorithms.

In order to confirm that the timing results matched our complexity analysis results, we

first graphed the asymptotic complexities with the timing results. When graphing, we multiplied constants to the asymptotic complexity to match the starting points of the complexity graph and test result graph because asymptotic complexity is computed with O which eliminates any constant multiple. Since the complexities are computed for the worst cases, they are not expected to be exactly the same as the test results, but their graphs are expected to have similar shapes to the graphs of test results. For example, if the complexity of an algorithm is exponential in n, the timing results of the algorithm would be also exponential in n, not constant, linear, or doubly exponential. In addition, we also performed an analysis for comparing the complexities and the test results of any two algorithms when both algorithms were run on more than 5 data sets. We took quotients of the complexities and the test results of the two algorithms to compute the ratios and plot the ratios in the same figure to see how they compare. All the asymptotic complexities are in Table 4.1 so we did not repeat them in this chapter.

We begin with a summary of observations from the timing tests.

- In all test cases, for the same norm equation, Algorithm 12 outperformed Algorithm 10 and Algorithm 11.
- The timing results are largely well aligned with our complexity analysis results.
- Using compact representations is not only space efficient, but also time efficient in practice. Algorithm 11 outperformed Algorithm 10 in all test cases, except for some cases with minimal parameters.
- With a set of minimal parameters, $g = 1$, $q = 3$, $n = 1$, $\deg c = 1$, Algorithm 10 was as fast as Algorithm 11 (see Figure 5.8, Figure 5.1, and Figure 5.10).
- The existing algorithm, Algorithm 10 became slow as any of n, q, g, or $\deg c$ grows, much quicker than the other algorithms, as expected from the complexity analysis in Section 3.1.2. In some tests, it was not feasible to time Algorithm 10 even with relatively small parameters.

- As expected from our complexity analyses, the number of factors of c has little impact on the performance of Algorithm 10, but it greatly affects Algorithm 11 and Algorithm 12.

- The timing results of Algorithm 11 and Algorithm 12 varied greatly when n or the number of factors of c was large, because these quantities determine the range of the number of ideals above cO_F.

- The average CPU time taken by computing an S-unit value matrix was negligible compared to the time of Algorithm 10 and Algorithm 11 in all test examples. However, computing an S-unit value matrix took longer than Algorithm 12 in many cases.

Now we discuss the individual timing results in detail. Varying the parameters related to F is first considered. The parameters are the degree n, the genus g of F and the size of the base field q, and the results are given in Sections 5.1, 5.2, and 5.3, respectively. Then we consider varying parameters related to c, which are $\deg(c)$ and the number of distinct factors of c. The results are presented in Sections 5.4 and 5.5, respectively.

5.1 Varying n

This section includes the timing results with g, q, $\deg c$ fixed, and with n varied. Since Algorithm 10 was very sensitive to a small change in parameters, three tests were conducted; one with minimal fixed parameters for which Algorithm 10 was too slow that included Algorithm 10 and the others with bigger fixed parameters to compare the behavior of Algorithm 11 and Algorithm 12. Figure 5.1 shows the result of the former and Figure 5.3 and Figure 5.4 show the result of the latter. The comparison between the testing results of Algorithm 11 and Algortihm 12 is provided in Figure 5.2.

Figure 5.1 provides the timing results of Algorithm 10, Algorithm 11, and Algorithm 12 with fixed $g = 1$, $q = 3$, $\deg c = 1$, along with the asymptotic complexities of them. Note

that the y-axis of Figure 5.1 is log scaled. The timing data of Algorithm 10 are only available for $n = 2, 4, 5$, because when $n = 7$, solving one norm equation took more than a day due to a huge search space. $n = 3$ and $n = 6$ were not tested because the greatest common divisor of n and q is not 1 for those n. Each algorithm performed better than what was expected from its asymptotic complexity because the asymptotic complexities are computed for the worst cases. In particular, the shapes of the graphs of the asymptotic complexities and timing results are largely similar, but the growth rates of the running time in practice is less than those of the asymptotic complexities.

Algorithm 11 was faster than Algorithm 10 in most of the testing samples, and Algorithm 12 was the fastest in all the testing samples. There were some samples where Algorithm 10 was faster than Algorithm 11 when $n = 2$, but on average Algorithm 11 was faster than Algorithm 10. From $n = 3$ on, Algorithm 11 started to outperform Algorithm 10 greatly. Algorithm 10 and Algorithm 11 are affected by the growth in n, mainly because their search spaces expand exponentially as n grows, while Algorithm 12 is affected because of the number of ideals to search and the cost of searching for a close vector and the number of ideals above cO_F.

Due to limited data, we did not compare Algorithm 10 with the other algorithms by comparing the quotients of asymptotic complexities and timing results. However, we confirmed that Algorithm 10 tends to have exponentially more elements in its search space compared to Algorithm 11 as we expected from the complexity analysis. For example, when $n = 7$, an example had $3^{21} (\approx 2^{33})$ elements in the search space of Algorithm 10, while the same example has only 1 element in the search space of Algorithm 11.

Figure 5.2 shows the quotients of the timing results of Algorithm 11 and Algorithm 12, and the quotients of the asymptotic complexities of the algorithms. They roughly have similar shapes, meaning that Algorithm 12 is faster than Algorithm 11 in practice roughly by a factor of $O(2^{x^3/4-4})$, which is the dominating factor of the quotient of their asymptotic complexities with the fixed parameters.

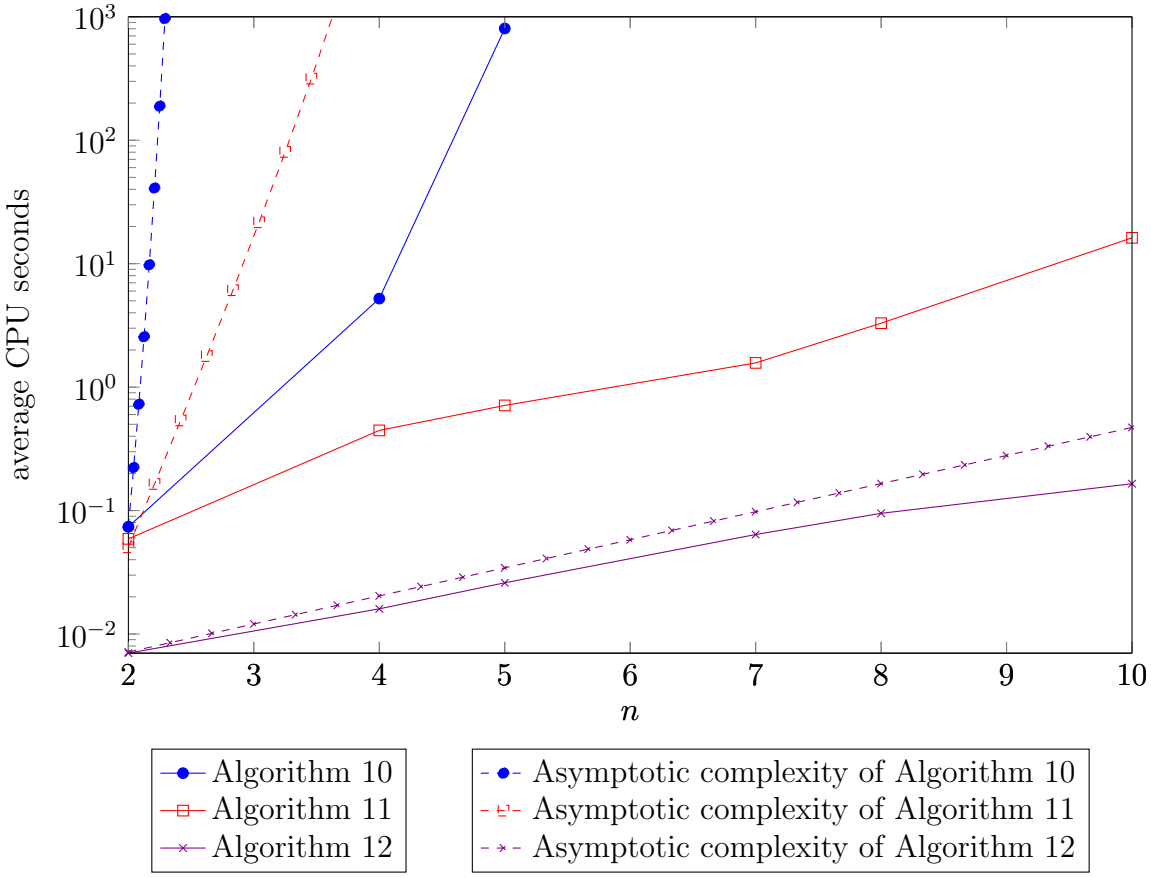

Figure 5.1: Empirical timing results and asymptotic complexities for varying n ($g = 1$, $q = 3$, $h_{O_F} = 1$, $\deg c = 1$, and irreducible c)

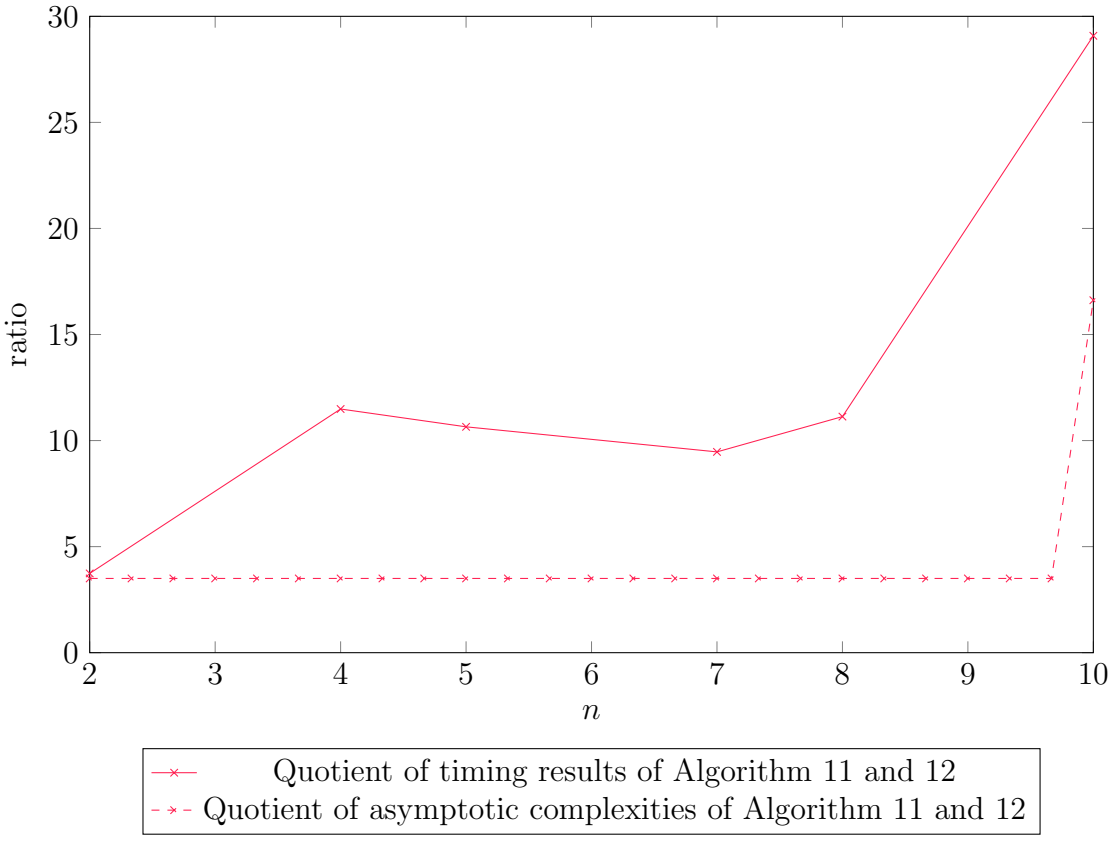

Figure 5.2: Quotient analysis of Algorithm 11 and Algorithm 12 for varying n ($g = 1$, $q = 3$, $h_{O_F} = 1$, $\deg c = 1$, irreducible c)

Figure 5.3 includes the timing results with bigger fixed parameters, $g = 3$, $q = 17$, $\deg c = 1$, along with the graphs of their asymptotic complexities. Note that the y-axis of only the upper graph of Figure 5.3 is log scaled. Algorthm 11 and Algorithm 12 are plotted separately due to the different ranges of values in the data. Figure 5.3 includes the timing results of Algorithm 11 up to $n = 5$. When $n = 6$, Algorithm 11 started to take more than a day to solve a norm equation, but Algorithm 12 took 1.735 CPU seconds on average. In Figure 5.3, Algorithm 11 grows exponentially faster than Algorithm 12, due to the number of elements in the search space of Algorithm 11 and the number of ideals to be tested in Algorithm 12, which were analyzed as $O(n^n 2^{n^3/4+n\deg c} q^{gn})$ and $O(2^{n\deg c})$, respectively, in our complexity analysis.

Though the graphs of Algorithm 11 and Algorithm 12 generally have similar shapes with the graphs of their asymptotic complexities in Figure 5.3, the test result graphs do not grow as much as the complexity graphs because the dependence on n in the complexity arises from the number of ideals above cO_F or the unit rank of O_F, which can vary for each example and can not be bigger than n.

An extra timing test for Algorithm 12 with even bigger fixed parameters was conducted to investigate how the growth of n affects the average time Algorithm 12 takes to solve a norm equation. We set the fixed parameters $g = 3$, $q = 17$, $\deg c = 10$, and the number of distinct factors of c was 10. Figure 5.4 shows the timing results of the test along with the graph of the asymptotic complexity of Algorithm 12. Note that the y-axis is log scaled. Under this parameter setting, Algorithm 12 started to take more than a day to solve a norm equation when $n = 8$. As expected from the complexity analysis, the run time of Algorithm 12 grew exponentially as n grows, but it grew less than the complexity because the complexity is computed for the worst case.

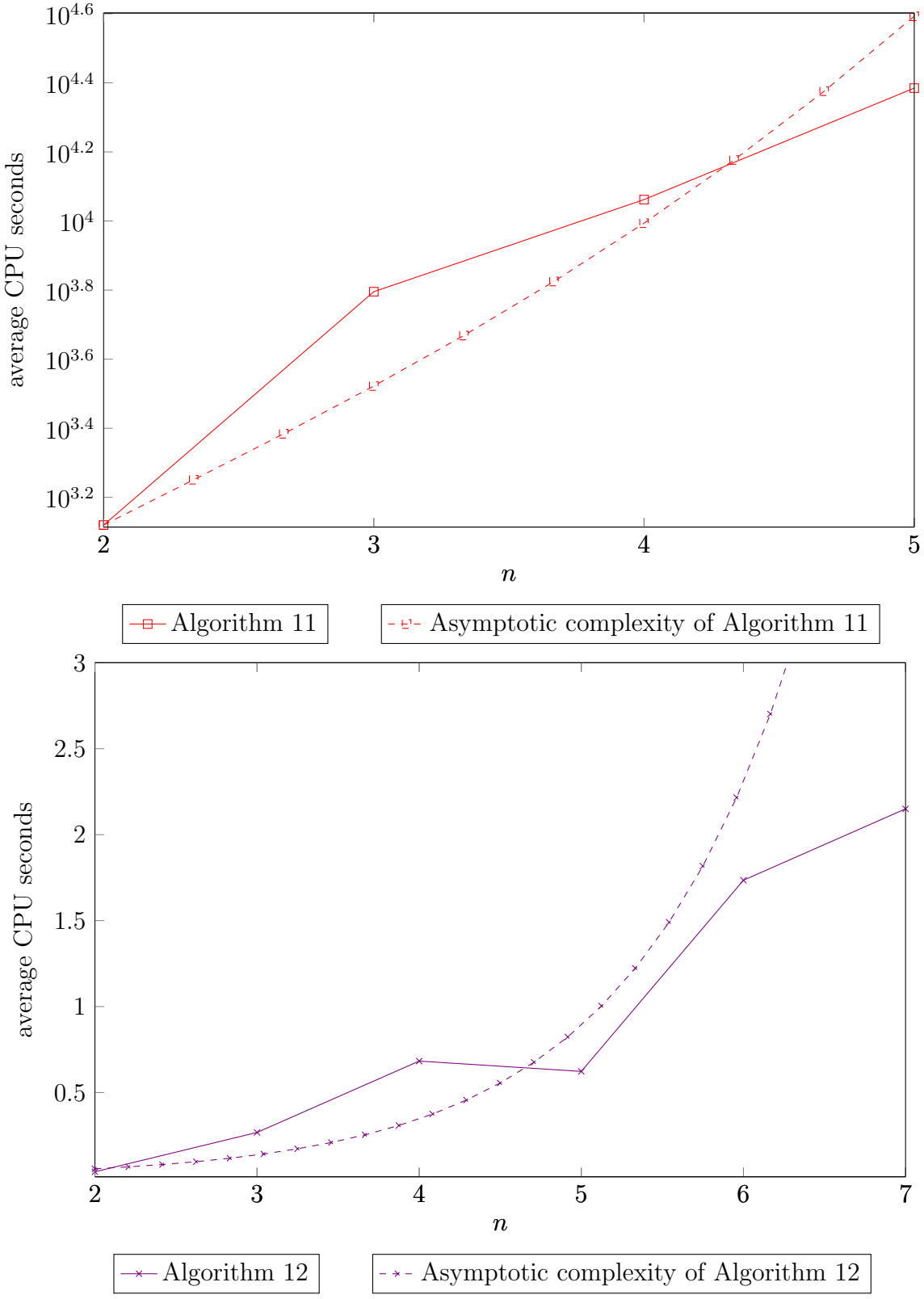

Figure 5.3: Empirical timing results and asymptotic complexities of Algorithm 11 and Algorithm 12 for varying n ($g = 3$, $q = 17$, $h_{O_F} = 1$, $\deg c = 1$, irreducible c)

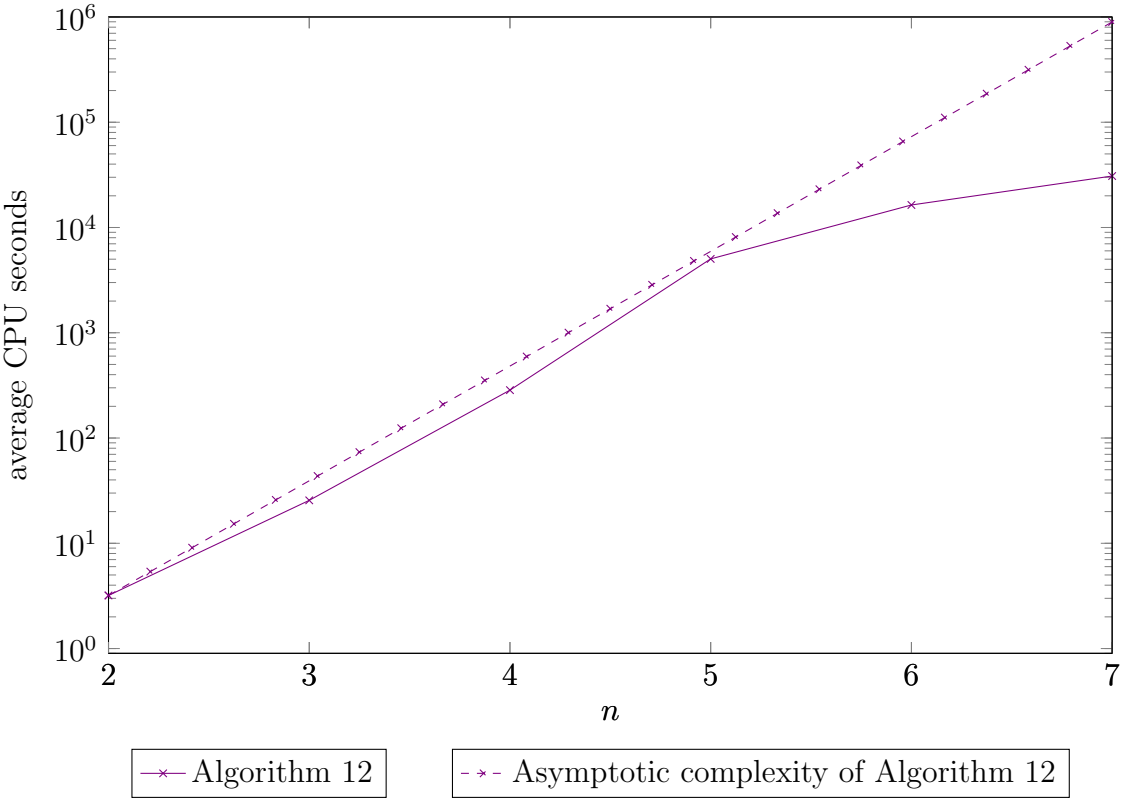

Figure 5.4: Empirical timing result and asymptotic complexity of Algorithm 12 for varying n ($g = 3$, $q = 17$, $h_{O_F} = 1$, $\deg c = 10$, and c has 10 distinct linear factors.)

5.2 Varying g

In this section, we discuss our timing results for varying g, while n, q, $\deg c$ and the number of distinct factors of c are fixed. Since Algorithm 10 and Algorithm 11 are more sensitive to a small change in g than Algorithm 12, two tests were conducted; one with minimal fixed parameters and timed for all three algorithms and the other one with bigger fixed parameters and with g increased further to see the performance of Algorithm 12. Figure 5.5 shows the result of the former and Figure 5.7 shows the result of the latter. Figure 5.6 shows the comparison between the test results of Algorithm 11 and Algortihm 12.

Figure 5.5 shows the timing results with minimal fixed parameters, $n = 2$, $q = 3$, $\deg c = 1$, along with the graphs of the asymptotic complexities as functions in g. The y-axis of Figure 5.5 is log scaled. The graph of the test results has a similar shape to the graph of the complexities, which means the test results were as expected from our complexity analysis. Though Algorithm 10 was timed, the timing result of it is not plotted in Figure 5.5 because the test had to be forcefully terminated. The only data collected for Algorithm 10 is that it took 0.059 CPU seconds on average when $g = 1$. When $g = 2$, Algorithm 10 did not finish running one example for more than 3 days, which is significantly longer than the other algorithms and exceeds the time frame that we set at the beginning of this chapter. This was mainly because the search space of Algorithm 10 grows doubly exponentially, which was expected from our complexity analysis. The number of elements in the search space from our complexity analysis is asymptotically $O\left(q^{n^2 2^{n^2/4} q^g}\right)$. In practice, with our fixed parameters, when $g = 1$, the number of elements in a search space was around $3^6 (\approx 2^{10})$, but when $g = 2$, it was $3^{20} (\approx 2^{32})$. Figure 5.5 includes data for Algorithm 11 for g up to 8, because when $g = 9$, Algorithm 11 started to take more than a day to solve a norm equation. It was also the number of elements in the search space of Algorithm 11 that makes the algorithm slow because the number increases exponentially as g grows. For example, for a fixed c, there were 11 elements in the search space for a function field F of genus $g = 1$, 65 elements for F of $g = 2$, 131 elements for F of $g = 3$, and 213 elements for F of $g = 4$. The number

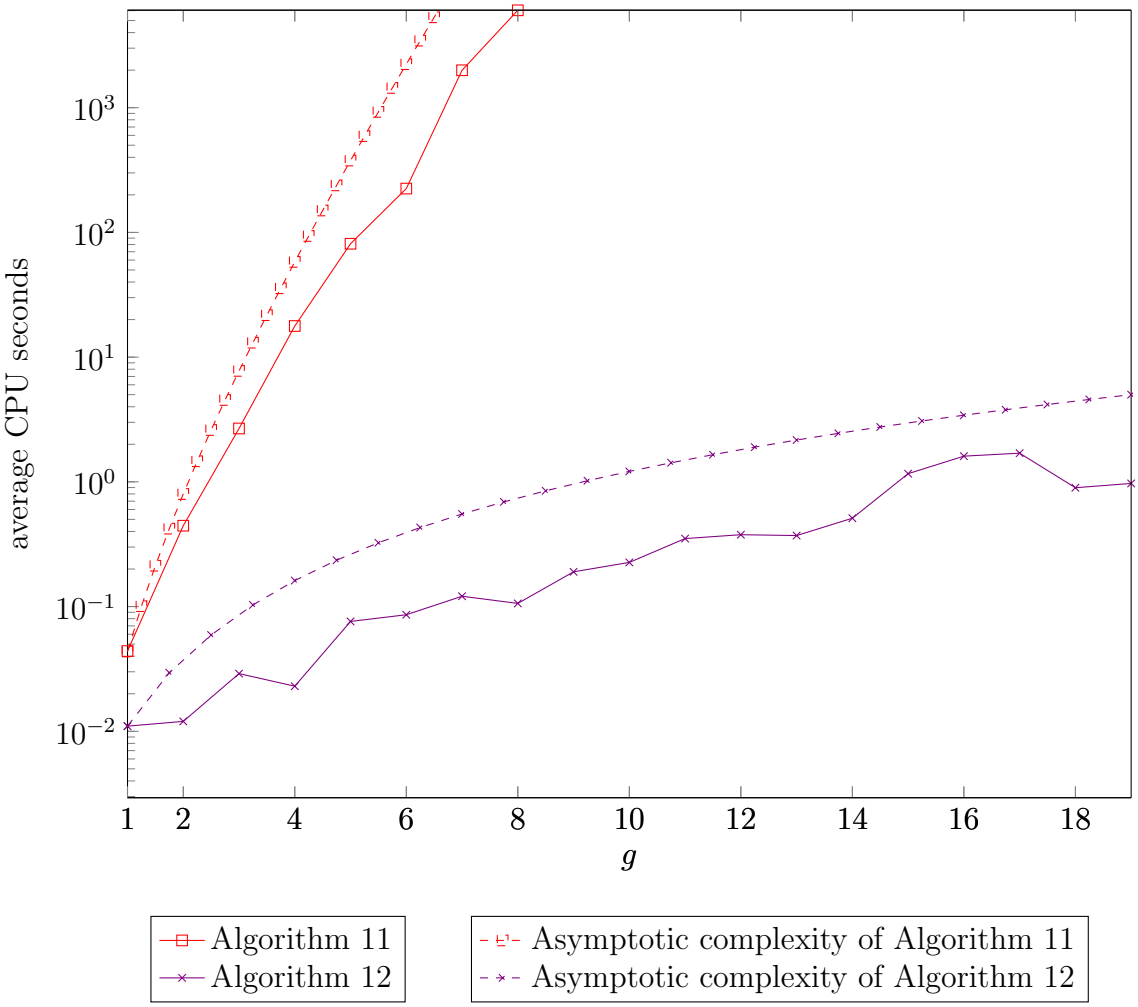

Figure 5.5: Empirical timing results and asymptotic complexities for varying g ($n = 2$, $q = 3$, $h_{O_F} = 1$, $\deg c = 1$, irreducible c)

of elements can vary in different examples because it is not solely determined by g and the fixed parameters but is affected by the unit rank of F, the regulator, and the number of places above c. Algorithm 12 was the fastest in all test examples. In this setting, computing the value matrix using Hess's relation search algorithm took more time than Algorithm 12.

A quotient analysis of the test results and the asymptotic complexities of Algorithm 11 and Algorithm 12 was conducted. Figure 5.6 includes the graphs of the ratios of the test results and the complexities of the algorithms. The graphs show that the test results are as expected from our complexity analysis results.

An extra test was conducted for Algorithm 12 with bigger fixed parameters. Figure 5.7

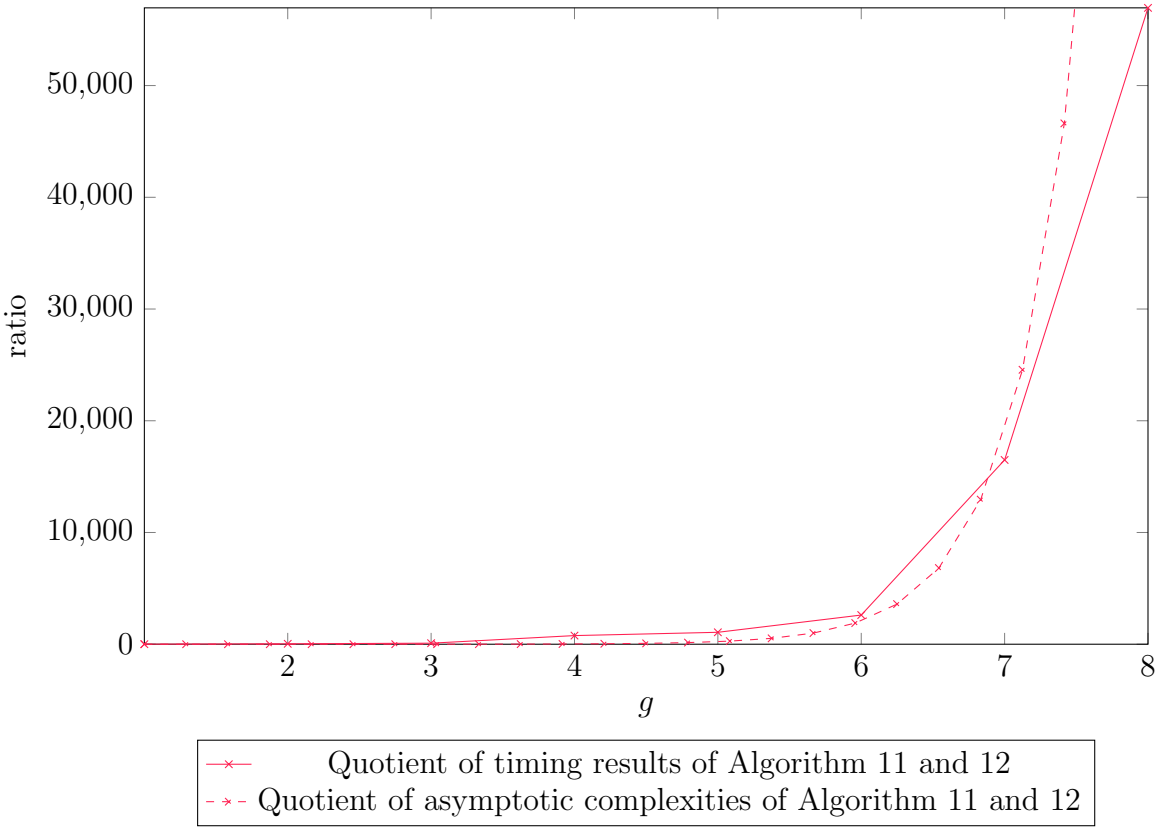

Figure 5.6: Quotient analysis of Algorithm 11 and Algorithm 12 for varying g ($n = 2$, $q = 3$, $h_{O_F} = 1$, $\deg c = 1$, irreducible c)

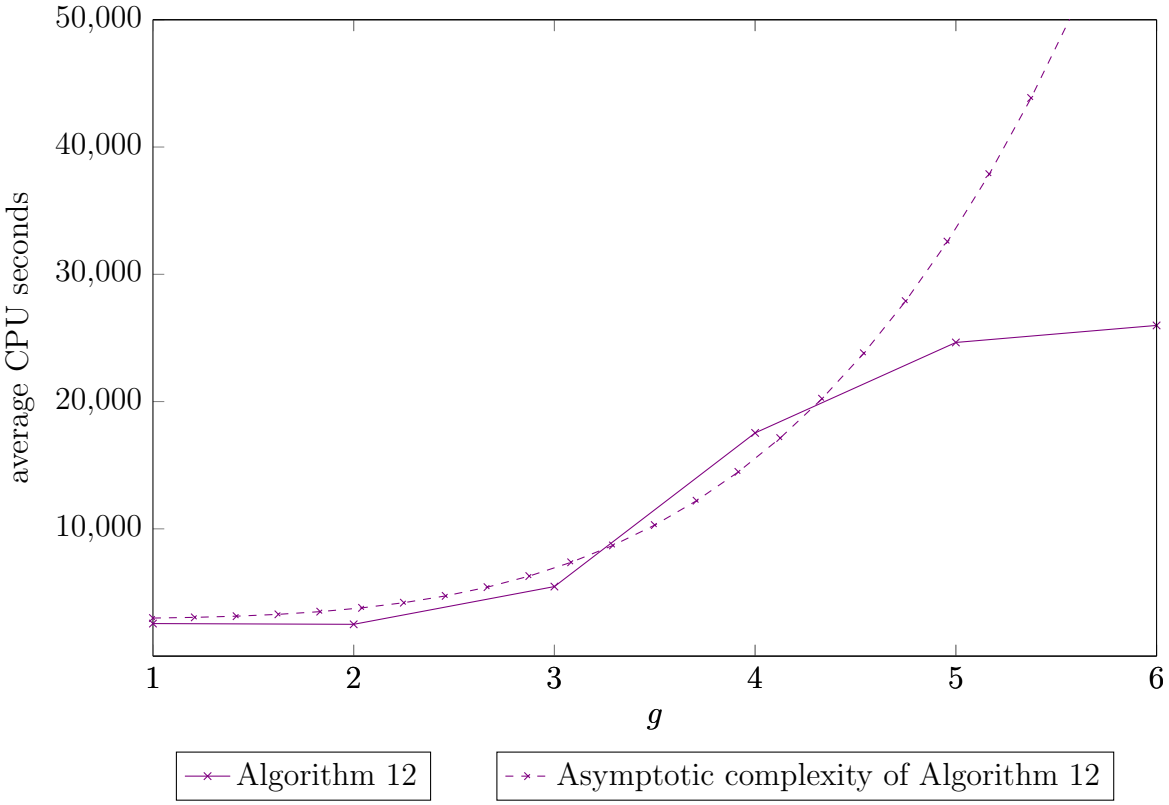

Figure 5.7: Empirical timing result and asymptotic complexity for varying g of Algorithm 12 ($n = 2$, $q = 8191$, and $\deg c = 1$, irreducible c)

shows the timing results of Algorithm 12 with $n = 2$, $q = 8191$, and $\deg c = 1$ and the asymptotic complexity of the algorithm. In this test, we used F with arbitrary ideal class number as Algorithm 12 has the same asymptotic complexity for F with any ideal class number. The timing results generally align well with our complexity analysis. Computing the value matrix using Hess's relation search algorithm took longer than Algorithm 12, which is polynomial in g. This is as expected from our complexity analysis because computing the value matrix is subexponential in g from Lemma 2.38.

5.3 Varying q

In this section, we described our experiments for the algorithms for varying q while g, n, $\deg c$, and the number of distinct factors of c are fixed. One test was conducted with minimal

fixed parameters.

Figure 5.8 shows the result of the test with the fixed parameters $g = 1$, $n = 2$, and $\deg c = 1$, and the graphs of the asymptotic complexities of the algorithms. Note that the y-axis of Figure 5.8 is log scaled. Timing data for Algorithm 10 is available only for $q = 3$ and 5, because when $q = 7$, solving one norm equation already took more than 2 days on average, which is significantly longer than the other algorithms and exceeds the time frame we set at the beginning of this chapter. Therefore, Algorithm 10 was not timed with larger q. On the other hand, Algorithm 11 and Algorithm 12 were timed for up to a 13-bit prime q and a 14-bit prime q, respectively. They started to take more than a day to solve a norm equation, including precomputations, when q is a 14-bit prime and a 15-bit prime, respectively. As seen in Figure 5.8, Algorithm 11 and Algorithm 12 outperformed Algorithm 10 for all q. When $q = 3$, Algorithm 10 solved norm equations almost as fast as Algorithm 11. There were some norm equations that Algorithm 10 solved faster than Algorithm 11, but on average, Algorithm 11 was faster than Algorithm 10.

Figure 5.8 also shows that the test results are aligned well with the corresponding complexity analysis results. The asymptotic complexity graph of Algorithm 10 grows faster in Figure 5.8 than those of Algorithm 11 and Algorithm 12. This is mainly because the number of elements in the search space of Algorithm 10 grows superexponentially as q grows, unlike the other algorithms.

Although each example has a different size of search space due to a different unit rank and regulator, we confirmed that the number of elements in the search space is largely as we expected from the complexity analysis. For example, with our fixed parameters when $q = 7$, there were $7^{12} (\approx 2^{34})$ elements in the search space, while there were $5^{10} (\approx 2^{23})$ elements in the search space when $q = 5$. Algorithm 11 is affected by the growth of q, also because of the number of elements in the search space, which is polynomial in q, $O(q^{gn})$. The growth of q has the least effect on the performance of Algorithm 12, as the number of ideals to test in this algorithm does not depend on q. In our test, Algorithm 11 and Algorithm 12 took

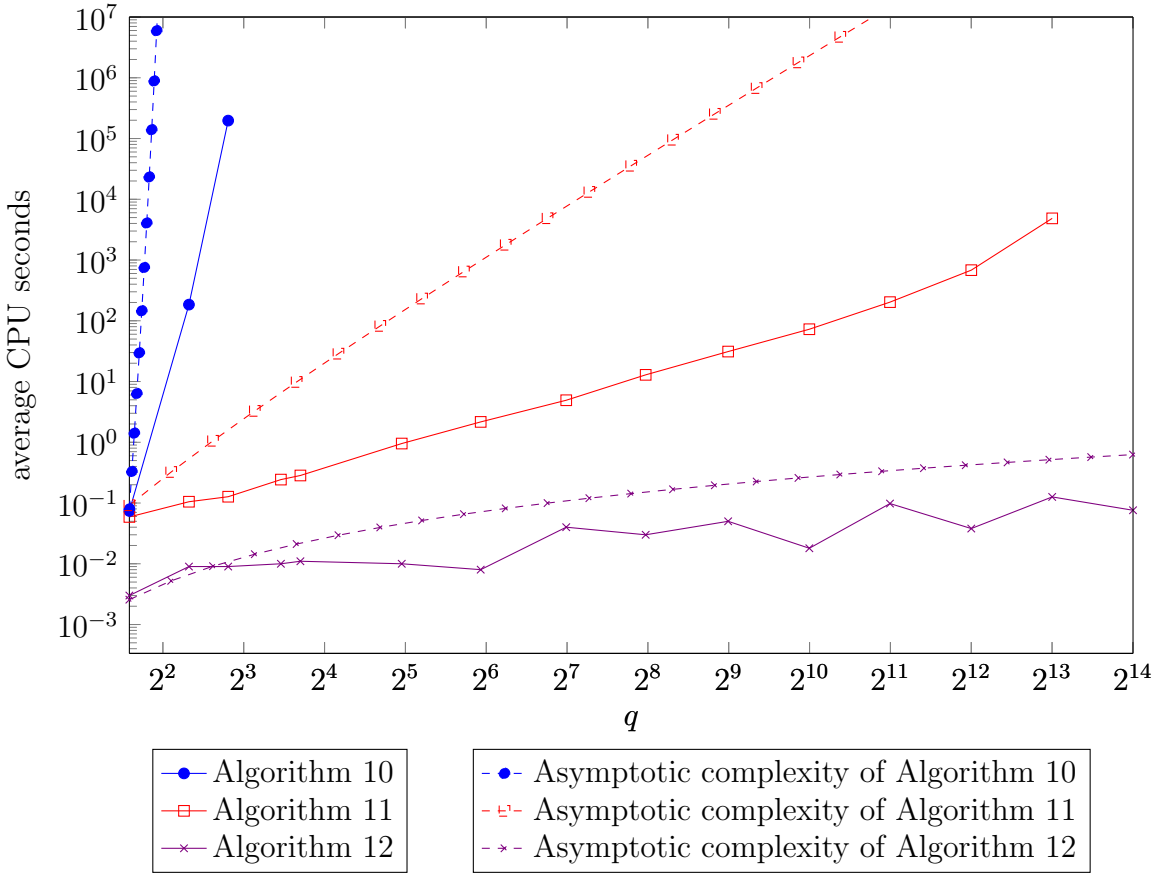

Figure 5.8: Empirical timing results and asymptotic complexities for varying q ($g = 1$, $n = 2$, $h_{O_F} = 1$, $\deg c = 1$, irreducible c)

less time than the precomputation, computing the value matrix, and show similar shapes in Figure 5.8. This result matches our complexity analysis that the cost of precomputation is larger than the costs of Algorithm 11 and Algorithm 12, which are polynomial in q and polynomial in $\log q$, respectively, as in Table 4.1.

Next, we compared Algorithm 11 and Algorithm 12 by taking quotients of the testing data and their complexities. The ratios of test data and complexities are in Figure 5.9. Note that both the x and y axes of Figure 5.9 are log scaled. Due to the nature of random examples, the quotient of the timing results graph is not as smooth as the complexity graph, but their shapes are largely aligned.

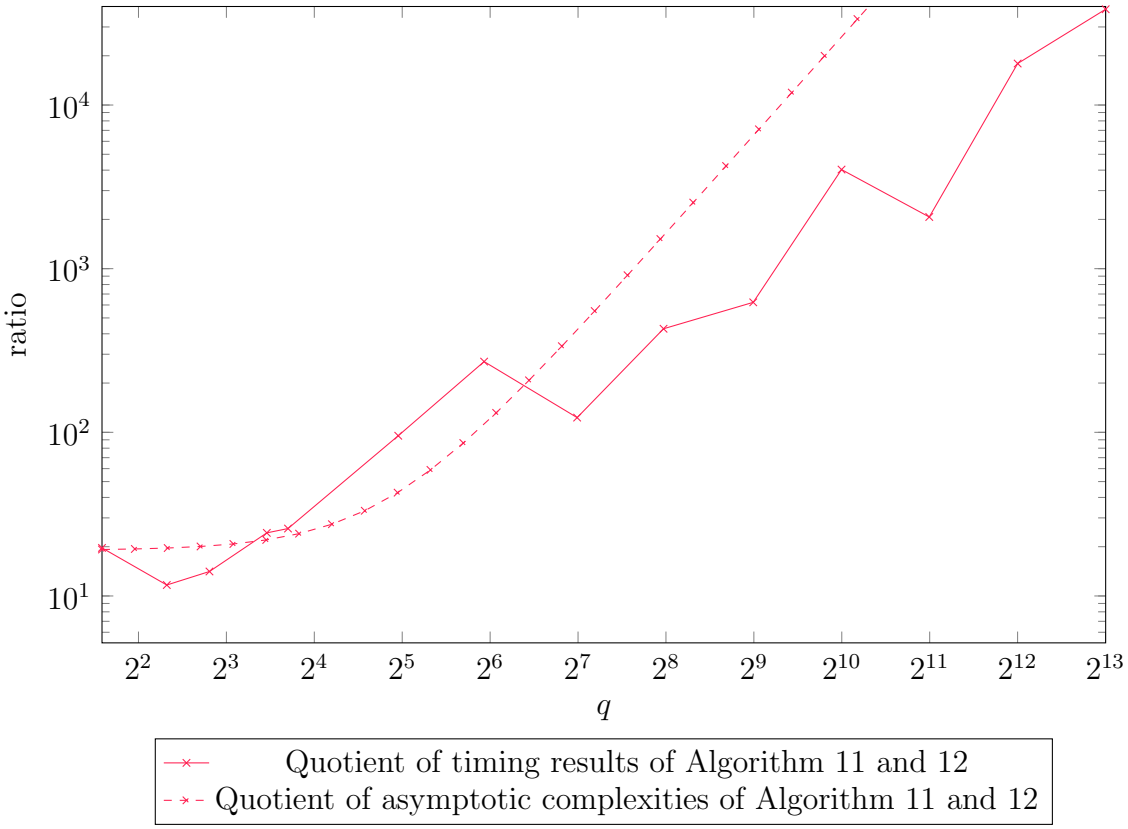

Figure 5.9: Quotient analysis of Algorithm 11 and Algorithm 12 for varying q ($g = 1$, $n = 2$, $h_{O_F} = 1$, $\deg c = 1$, irreducible c)

5.4 Varying $\deg(c)$

In this section, we describe our tests of the algorithms for only varying $\deg c$, while g, n, q, h_{O_F}, and the number of factors of $\deg c$ are fixed. In all test cases, irreducible c were chosen. Two tests were conducted varying $\deg c$; one with minimal fixed parameters to time all three algorithms and the other one with bigger fixed parameters to time Algorithm 11 and Algorithm 12 only. Figure 5.10 shows the timing results of the former, and Figure 5.12 shows the result of the latter. In all test cases, the algorithms with compact representations, Algorithm 11 and Algorithm 12, were faster than the existing algorithm, Algorithm 10, and Algorithm 12 was the fastest.

Figure 5.10 provides the timing results of all three algorithms with minimal fixed parameters, $g = 1$, $n = 2$, $q = 3$, along with the graphs of their asymptotic complexities. Note that Figure 5.10 has its y-axis log scaled. When $\deg c = 13$, Algorithm 10 took more than a day to solve a norm equation on average, thus Algorithm 10 was not timed for bigger $\deg c$.

Unlike the asymptotic complexity, the data of Algorithm 10 shows a "jump" for every n as $\deg c$ increases because the degree bounds that define the search space of Algorithm 10 involve $\frac{\deg c}{n}$ by Lemma 3.3. In the computation of the overall cost of Algorithm 10, the denominator n in $\frac{\deg c}{n}$ was canceled by another factor n, so the complexity graph does not show such jumps. The test results of the other algorithms did not show such jumps and exhibited largely similar shapes to their complexity graphs. As expected from the complexity analysis, Algorithm 10, whose run time is exponential in $\deg c$, grew faster in Figure 5.10 than Algorithm 11 and Algorithm 12, which are polynomial in $\deg c$. The graphs of Algorithm 11 and Algorithm 12 have similar shapes, which is also expected from our asymptotic complexity analysis which shows that Algorithm 11 is $O((\deg c)^\omega)$ and Algorithm 12 is $O((\deg c)^2)$, where ω is the matrix multiplication exponent, $\omega < 2.37286$ [2]. Algorithm 12 was constantly faster than Algorithm 11. This is because the non-dominating computations of Algorithm 12 are faster than those of Algorithm 11.

We took quotients of the test results of the algorithms and compared them to quotients

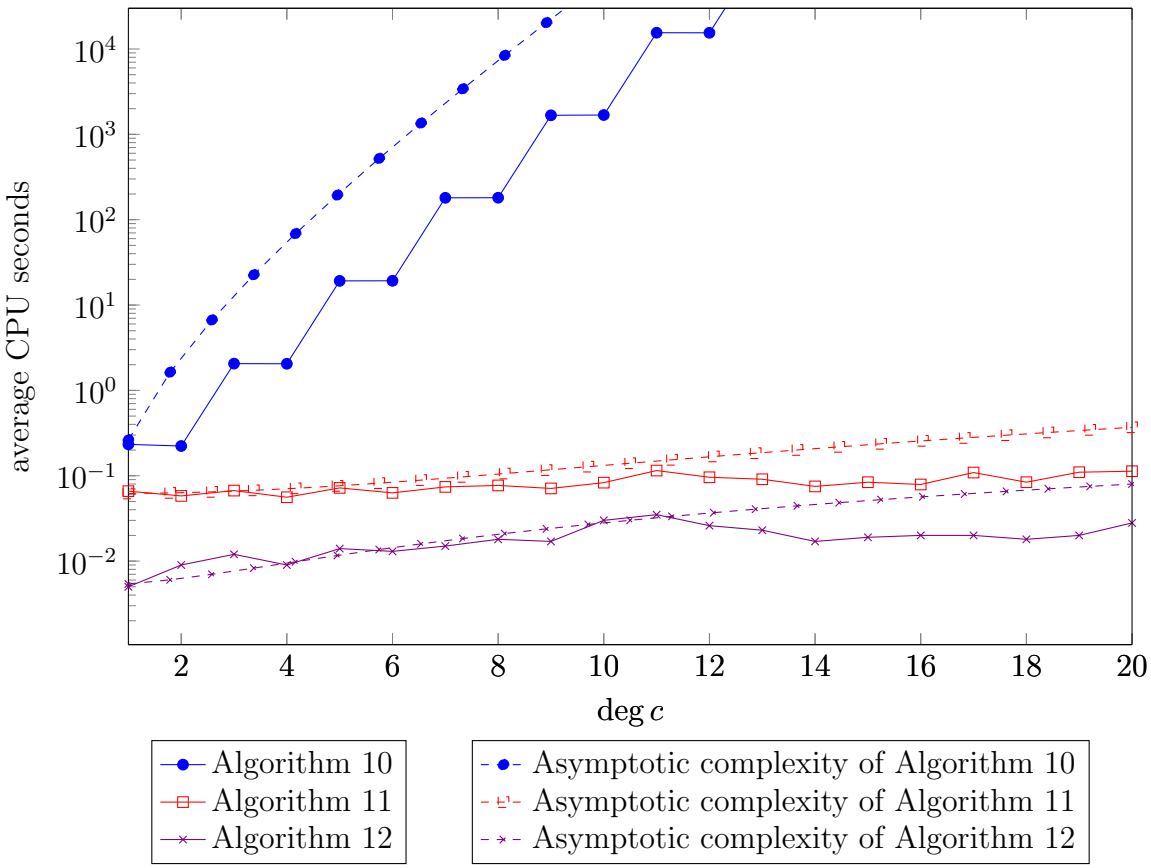

Figure 5.10: Empirical timing results and asymptotic complexities for varying $\deg c$
($g = 1$, $n = 2$, $q = 3$, $h_{O_F} = 1$, irreducible c)

of the complexities. The upper graph of Figure 5.11 compares Algorithm 10 to the other algorithms, and the lower graph compares Algorithm 11 and Algorithm 12. Both of them show that the differences between the algorithms are as expected from the complexity analysis. In particular, Algorithm 10 is exponentially slower than the other two, and the difference between Algorithm 11 and Algorithm 12 is polynomial in $\deg c$.

Figure 5.12 shows the timing results of Algorithm 11 and Algorithm 12 with slightly bigger fixed parameters, $g = 1$, $n = 3$, $q = 53$ and the graphs of the asymptotic complexities of the algorithms. With this parameter setting, Algorithm 11 started to take longer than a day to solve a norm equation when $\deg c$ is a 12-bit integer, while Algorithm 12 did when $\deg c$ is 15-bit. Similar to what we observed in Figure 5.10, Algorithm 11 and Algorithm 12 have similar shapes in graphs that are aligned well with their asymptotic complexities, and Algorithm 12 was faster than Algorithm 11 in all testing examples.

5.5 Varying the number of distinct factors of c

In this section, we present the results of the timing tests with variations in the number of distinct factors of c and without variation in g, n, q, h_{O_F}, and $\deg c$. From the complexity analyses on Algorithm 11 and Algorithm 12, we see that the number of distinct factors of c affects the asymptotic complexities of our new algorithms, Algorithm 11 and Algorithm 12. We conducted two tests with different fixed parameters; one with small fixed parameters to test all algorithms together, and the other with relatively larger parameters to time Algorithm 11 and Algorithm 12 only. Figure 5.13 and Figure 5.14 show the results of the tests. Note that the number of factors is trivially bounded by $\deg c$, and our timing tests were conducted with fixed $\deg c$. That means we can not consider the asymptotic complexities of the algorithms when only the number of distinct factors of c goes infinity while n, g, q, and $\deg c$ are fixed. Thus, in this section, we only present test results and do not compare them to asymptotic complexities.

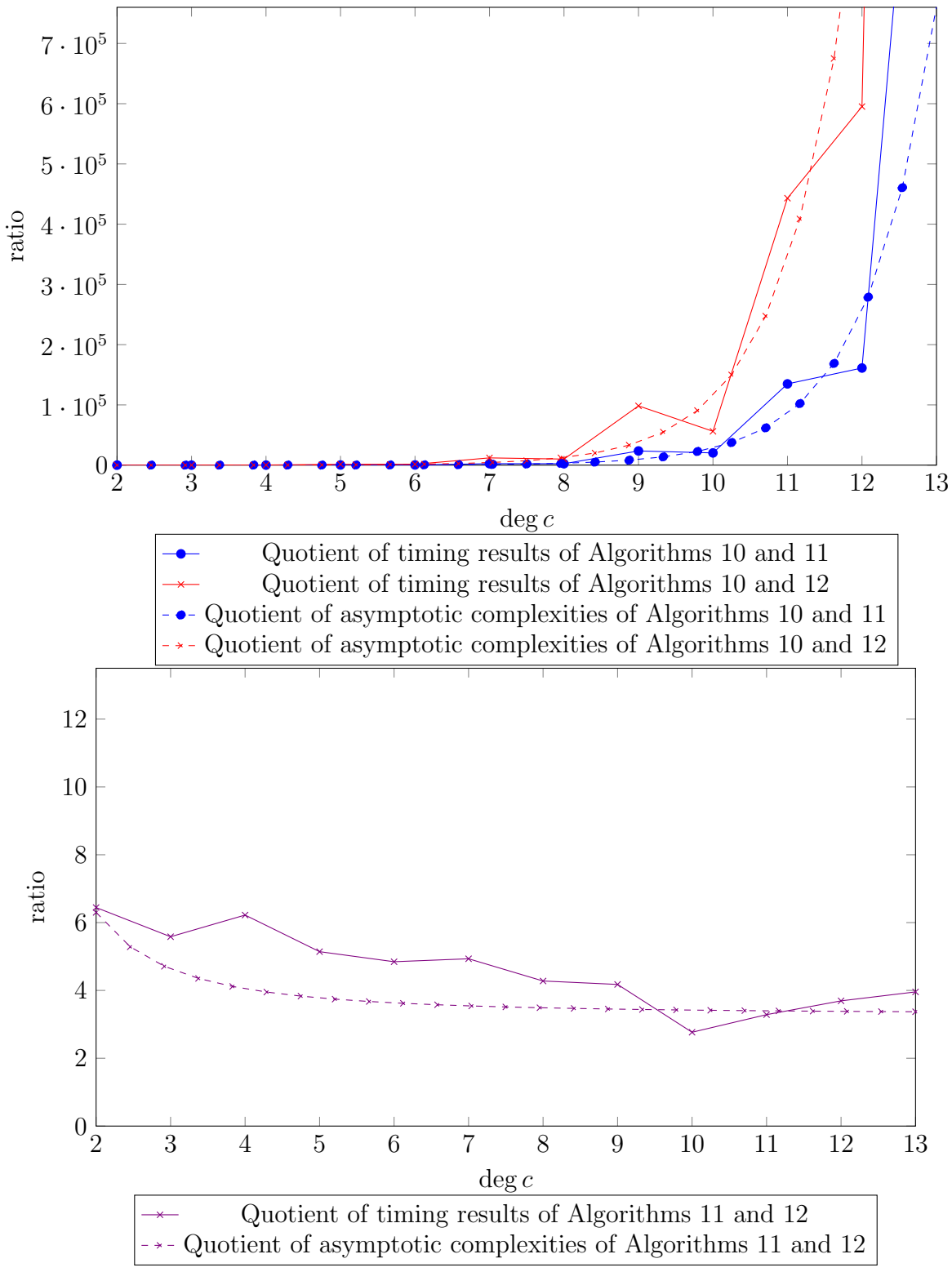

Figure 5.11: Quotient analysis of the algorithms for varying deg c ($g = 1$, $n = 2$, $q = 3$, $h_{O_F} = 1$, irreducible c)

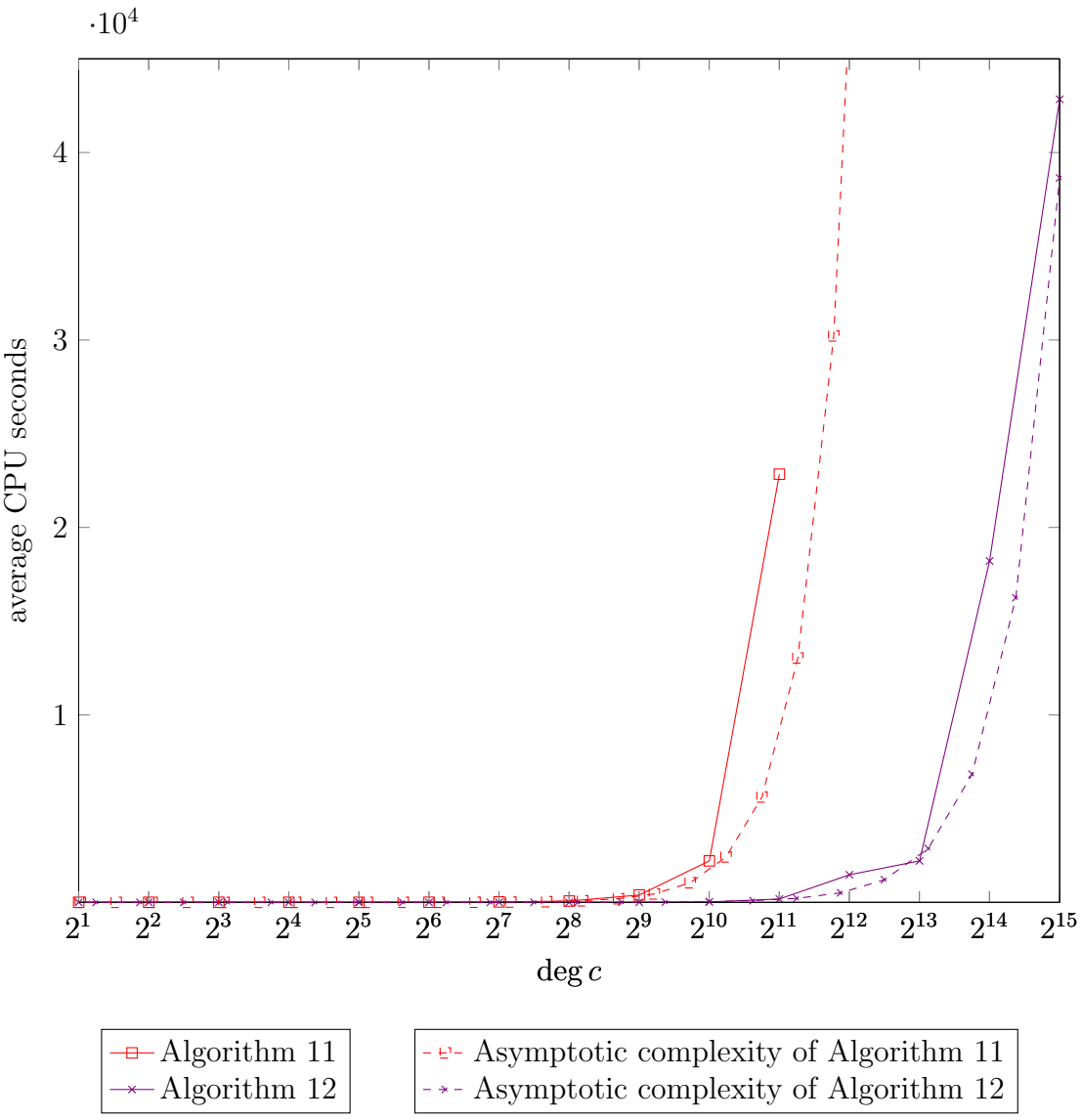

Figure 5.12: Empirical timing results and asymptotic complexities for varying $\deg c$ of Algorithm 11 and Algorithm 12 ($g = 1$, $n = 3$, $q = 53$, $h_{O_F} = 1$, irreducible c)

Figure 5.13 shows the timing results with small fixed parameters, $g = 1$, $n = 2$, $q = 3$, $\deg c = 10$. Since Algorithm 10 can be very slow even with relatively small parameters, the parameters for the test in Figure 5.13 were carefully chosen based on the results of the previous tests. Note that Figure 5.13 has a discontinuity in the y-axis due to the different ranges of values in the data. Figure 5.13 shows that a change in the number of distinct factors of c has little effect on the performance of Algorithm 10 as expected from the complexity analysis. Algorithm 12 was more tolerant to small changes in the number of factors of c than Algorithm 11. This result matches our complexity analysis that the asymptotic complexity and the non-dominating factors of the complexity of Algorithm 12 are also smaller than those of Algorithm 11.

Figure 5.14 provides the timing results of Algorithm 11 and Algorithm 12, which were timed with $g = 1$, $n = 3$, $q = 53$, and $\deg c = 50$. Note that the y-axis of Figure 5.14 is log scaled. The growth in the number of factors of c greatly affects the timing results of Algorithm 11 and Algorithm 12. The results are aligned with the complexity analysis results, that both algorithms are polynomial in $\deg c$ when c is irreducible, but they become exponential in $\deg c$ when c splits into linear factors. In Figure 5.14, the growth rate of the test results of Algorithm 12 grows as the number of distinct factors of c grows. Algorithm 12 took longer than the precomputation of computing the value matrix, when the number of distinct factors of c was more than 7 in this setting. Algorithm 11 started to take longer than a day to solve a norm equation when the number of distinct factors of c is 9, and Algorithm 12 did when the number of distinct factors of c is 14.

5.6 Summary

In this section, we summarize our observations and findings from the timing tests.

Overall, the empirical testing results are well aligned with our complexity analyses provided in Section 3.1.2, Section 3.2.2, and Section 4.1.2. Our new algorithms, Algorithm 11

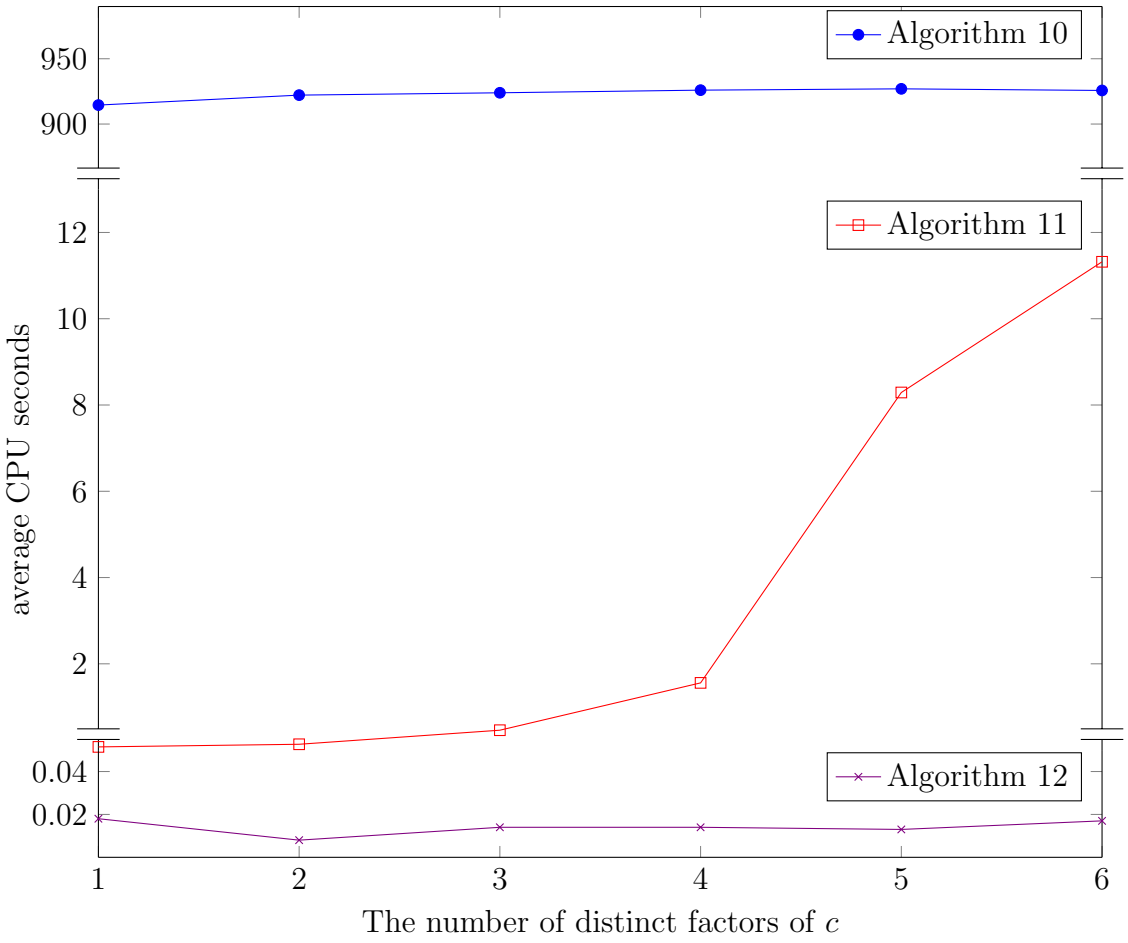

Figure 5.13: Empirical timing results for varying the number of distinct factors of c
($g = 1$, $n = 2$, $q = 3$, $h_{O_F} = 1$, $\deg c = 10$)

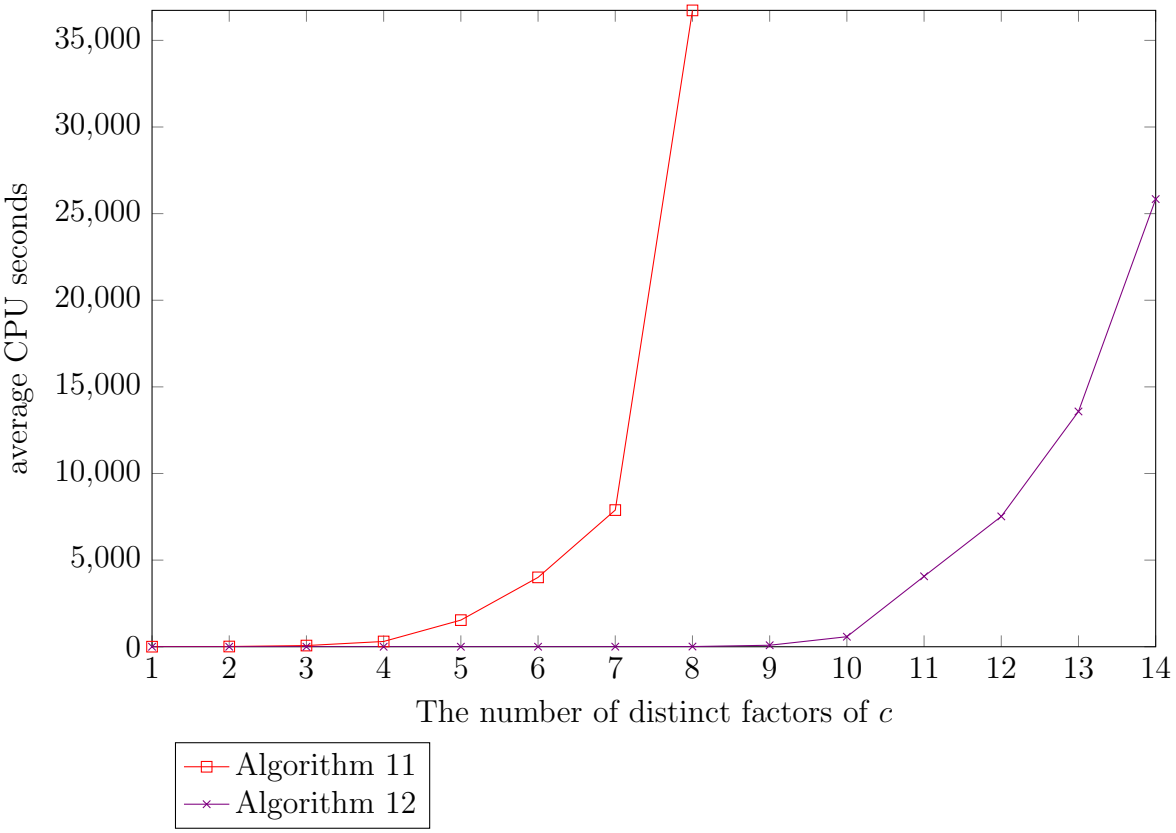

Figure 5.14: Empirical timing results of Aglorithm 11 and Algorithm 12 for varying the number of distinct factors of c
($g = 1$, $n = 3$, $q = 53$, $h_{O_F} = 1$, $\deg c = 50$)

and Algorithm 12, greatly outperformed the existing algorithm, Algorithm 10 in practice, as expected from the complexity analyses. Our test results show that Algorithm 12 is the fastest in all parameter settings.

When all parameters are small, all three algorithms solved norm equations quickly. That means Algorithm 10 can also be efficiently used when all parameters are minimal, and $g \leq 1$. However, Algorithm 10 became inefficient immediately once the parameters were slightly increased and were no longer minimal. This is because the number of elements in the search space of Algorithm 10 is doubly exponential in n and g, superexponential in q, and exponential in $\deg c$. The number of factors of c had little effect on the performance of the algorithm, as we expect from the complexity analysis.

Overall, Algorithm 11 outperformed Algorithm 10. As in our complexity analysis in Section 3.2.2, the number of elements in the search space of Algorithm 11 is superexponential in n, exponential in g and $\deg c$, and polynomial in q. Unlike Algorithm 10, the performance of Algorithm 11 was affected greatly by the number of distinct factors of c.

Algorithm 12 greatly outperformed the other algorithms in all test cases. In most cases, the precomputation of computing the value matrix using Hess's relation search algorithm took longer than Algorithm 12. Algorithm 12 took more time than the precomputation only when the number of distinct factors of c was more than 7 in the test whose results are provided in Figure 5.14, as the number of ideals to be tested in Algorithm 12 is highly dependent on the number of distinct factors of c.

Chapter 6

Conclusion

In this thesis, we developed two novel algorithms for solving norm equations. The algorithms are more time and space efficient than the existing method by Gaál and Pohst in [21] both in theory and in practice. Thorough complexity and empirical analyses of the new algorithms and the existing method were conducted and compared. The new algorithms and the existing algorithm for solving norm equations, and the algorithms for computing compact representations, and for performing arithmetic with compact representations were implemented in Magma, and their correctness was rigorously verified through black-box and white-box testing. Extensive numerical testing provided evidence of the efficiency of the novel algorithms. The Magma implementation, including code for all our numerical tests, can be found at https://github.com/s-leem/FF_NormEq_CR.

The first new algorithm was inspired by the Gaál-Pohst algorithm. The algorithm has a different search space and enumerates elements in the search space in compact representation, instead of the standard representation. Compared to Gaál-Pohst, this algorithm has a smaller search space, better asymptotic complexity, and smaller representations of solutions. In addition, this algorithm was faster, in practice, than the Gaál-Pohst algorithm on average in all tested parameter settings.

The second new algorithm searches for solutions by conducting a series of principal ideal

tests. For this algorithm, we designed a principal ideal test that involves a matrix equation. Similar to the first new algorithm, this algorithm has better asymptotic complexity, and smaller representations of solutions, compared to the Gaál-Pohst algorithm. In addition, the asymptotic complexity of this algorithm is even better than the first new algorithm. This was the fastest algorithm in practice in all tested inputs.

6.1 Future work

An open problem to extend this thesis is to investigate the performance of Algorithm 10 on norm equations over a global function field $F/\mathbb{F}_q$ with $\gcd(q, n) > 1$. Such F were excluded during the empirical analysis to avoid wild ramification. After running some examples over F with $\gcd(q, n) > 1$, it is noted that Algorithm 10 tends to have a larger search space for norm equations over such F compared to those with $\gcd(q, n) = 1$ that we used in our empirical analysis. It is unclear whether or not the larger search spaces are due to wild ramifications. Investigating the reasons may lead to developing efficient algorithms optimized for norm equations over those F.

As mentioned briefly in Section 1.1, a related problem of interest is to find solutions of norm equations in submodules M of O_F, instead of O_F. Unlike finding integral solutions, we do not consider solutions in M up to associates. This is because for an element $\alpha \in M$, an element that is associate to α is not guaranteed to be in M.

Regarding the algorithm for computing compact representations in [15], an open problem is to develop an algorithm computing compact representations that do not require F to have an infinite place of degree 1. An infinite place of degree 1 is required in Algorithm 4 to compute a correct compact representation of an element α, that is not just close to α. It is unclear how to ensure accuracy when there is no infinite place of degree 1.

Finally, from the successful result of this thesis, it seems promising to investigate different Diophantine equations over global function fields using compact representations. This leaves

us to explore related topics in the future.

Bibliography

[1] S. Alaca and K. S. Williams. *Introductory algebraic number theory*. Cambridge University Press, Cambridge, 2004.

[2] J. Alman and V. Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. [Society for Industrial and Applied Mathematics (SIAM)], Philadelphia, PA, 2021.

[3] J.-D. Bauch. *Lattices over polynomial rings and Applications to Function Fields*. PhD thesis, Universitat Autònoma de Barcelona, 2014.

[4] J.-D. Bauch. Genus computation of global function fields. *J. Symbolic Comput.*, 66:8–20, 2015.

[5] J.-D. Bauch. Lattices over polynomial rings and applications to function fields. *arXiv preprint* `arXiv:1601.01361 [math.NT]`, 2016.

[6] J.-D. Bauch, H. Tran, and S. Leem. Fast arithemtic in the divisor class group. *Unpublished manuscript*, 2020.

[7] J. Beck. Siegel's Lemma is sharp. In *A journey through discrete mathematics*, pages 165–206. Springer, Cham, 2017.

[8] A. Becker, N. Gama, and A. Joux. Solving shortest and closest vector problems: The decomposition approach. *IACR Cryptol. ePrint Arch.*, 2013:685, 2013.

[9] S. Birmpilis, G. Labahn, and A. Storjohann. Deterministic reduction of integer nonsingular linear system solving to matrix multiplication. In *ISSAC'19—Proceedings of the 2019 ACM International Symposium on Symbolic and Algebraic Computation*, pages 58–65. ACM, New York, 2019.

[10] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[11] J. Buchmann, C. Thiel, and H. Williams. Short representation of quadratic integers. In *Computational algebra and number theory (Sydney, 1992)*, volume 325 of *Math. Appl.*, pages 159–185. Kluwer Acad. Publ., Dordrecht, 1995.

[12] H. Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[13] P. M. Cohn. *Algebraic numbers and algebraic functions*. Chapman and Hall Mathematics Series. Chapman & Hall, London, 1991.

[14] C. Diem. An index calculus algorithm for plane curves of small degree. In *Algorithmic number theory*, volume 4076 of *Lecture Notes in Comput. Sci.*, pages 543–557. Springer, Berlin, 2006.

[15] K. Eisenträger and S. Hallgren. Computing the unit group, class group, and compact representations in algebraic function fields. In *ANTS X—Proceedings of the Tenth Algorithmic Number Theory Symposium*, volume 1 of *Open Book Ser.*, pages 335–358. Math. Sci. Publ., Berkeley, CA, 2013.

[16] C. Fieker, A. von Jurk, and M. E. Pohst. On solving relative norm equations in algebraic number fields. *Math. Comp.*, 66(217):399–410, 1997.

[17] U. Fincke and M. E. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Math. Comp.*, 44(170):463–471, 1985.

[18] F. Fontein. The infrastructure of a global field of arbitrary unit rank. *Math. Comp.*, 80(276):2325–2357, 2011.

[19] G. W. Fung and H. C. Williams. Compact representation of the fundamental unit in a complex cubic field. *Unpublished manuscript*, 1991.

[20] I. Gaál and M. E. Pohst. Diophantine equations over global function fields. I. The Thue equation. *J. Number Theory*, 119(1):49–65, 2006.

[21] I. Gaál and M. E. Pohst. On solving norm equations in global function fields. *J. Math. Cryptol.*, 3(3):237–248, 2009.

[22] I. Gaál and M. E. Pohst. Diophantine equations over global function fields. IV. S-unit equations in several variables with an application to norm form equations. *J. Number Theory*, 130(3):493–506, 2010.

[23] D. Harvey and J. van der Hoeven. Polynomial multiplication over finite fields in time $O(n \log n)$. *J. ACM*, 69(2):Art. 12, 40, 2022.

[24] F. Hess. Computing relations in divisor class groups of algebraic curves over finite fields. Available at http://www.staff.uni-oldenburg.de/florian.hess/publications/dlog.pdf.

[25] F. Hess. *Zur Divisorenklassengruppenberechnung in globalen Funktionenkörpern*. PhD thesis, Technical University Berlin, 1999.

[26] M. J. Jacobson, Jr., A. Pintér, and G. Walsh. A computational approach for solving $y^2 = 1^k + 2^k + \cdots + x^k$. *Math. Comp.*, 72(244):2099–2110, 2003.

[27] M. J. Jacobson, Jr. and H. C. Williams. Modular arithmetic on elements of small norm in quadratic fields. volume 27, pages 93–110. 2002. Special issue in honour of Ronald C. Mullin, Part II.

[28] M. J. Jacobson, Jr. and H. C. Williams. *Solving the Pell equation*. CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC. Springer, New York, 2009.

[29] G. Labahn, V. Neiger, and W. Zhou. Fast, deterministic computation of the Hermite normal form and determinant of a polynomial matrix. *J. Complexity*, 42:44–71, 2017.

[30] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[31] K. Mahler. Inequalities for ideal bases in algebraic number fields. *J. Austral. Math. Soc.*, 4:425–448, 1964.

[32] P. Mihăilescu. Primary cyclotomic units and a proof of Catalan's conjecture. *J. Reine Angew. Math.*, 572:167–195, 2004.

[33] M. E. Pohst and H. J. Zassenhaus. *Algorithmic algebraic number theory*, volume 30 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1989.

[34] M. Rosen. *Number theory in function fields*, volume 210 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002.

[35] R. Scheidler. Compact representation in real quadratic congruence function fields. In *Algorithmic number theory (Talence, 1996)*, volume 1122 of *Lecture Notes in Comput. Sci.*, pages 323–336. Springer, Berlin, 1996.

[36] R. Scheidler. Decision problems in quadratic function fields of high genus. *J. Complexity*, 16(2):411–423, 2000.

[37] W. M. Schmidt. Norm form equations. *Ann. of Math. (2)*, 96:526–551, 1972.

[38] J.-P. Serre. *Local fields*, volume 67 of *Graduate Texts in Mathematics*. Springer-Verlag, New York-Berlin, 1979. Translated from the French by Marvin Jay Greenberg.

[39] C. L. Siegel. Normen algebraischer Zahlen. *Nachr. Akad. Wiss. Göttingen Math.-Phys. Kl. II*, pages 197–215, 1973.

[40] A. K. Silvester, M. J. Jacobson, Jr., and H. C. Williams. Shorter compact representations in real quadratic fields. In *Number theory and cryptography*, volume 8260 of *Lecture Notes in Comput. Sci.*, pages 50–72. Springer, Heidelberg, 2013.

[41] D. Simon. Solving norm equations in relative number fields using S-units. *Math. Comp.*, 71(239):1287–1305, 2002.

[42] H. Stichtenoth. *Algebraic function fields and codes*, volume 254 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, second edition, 2009.

[43] H. P. F. Swinnerton-Dyer. *A brief guide to algebraic number theory*, volume 50 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 2001.

[44] A. Tang. *Infrastructure of function fields of unit rank one*. PhD thesis, University of Calgary, 2011.

[45] C. Thiel. Under the assumption of the generalized Riemann hypothesis verifying the class number belongs to NP$\cap$co-NP. In *Algorithmic number theory (Ithaca, NY, 1994)*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 234–247. Springer, Berlin, 1994.

[46] C. Thiel. *On the complexity of some problems in algorithmic algebraic number theory*. PhD thesis, Universitat des Saarlandes, Saarbrucken, Germany, 1995.

[47] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, third edition, 2013.

[48] A. Wiles. Modular elliptic curves and Fermat's last theorem. *Ann. of Math. (2)*, 141(3):443–551, 1995.

Appendix A

Examples of the standard representation and the compact representation

We provide examples of the representations of an element of a global function field. The standard representation is as defined in (2.29), and the compact representation is as defined in Definition 2.41.

Consider a function field $F = \mathbb{F}_5(x)(y)$ where y is a root of the polynomial

$$f(t) = t^3 + (x^3 + 1)t^2 + (x^5 + x^4 + 1)t + 2 \in \mathbb{F}_5[x][t].$$

The maximal order O_F of F is the $\mathbb{F}_5[x]$ module of rank 3,

$$O_F = \mathbb{F}_5[x] + y\mathbb{F}_5[x] + y^2\mathbb{F}_5[x].$$

Then $\mathcal{B} = \{1, y, y^2\}$ is an integral basis of F. The function field F has genus 6 and three infinite places $\{P_{1,\infty}, P_{2,\infty}, P_{3,\infty}\}$. A fundamental unit ε of F in the standard representation with respect to $\mathcal{B}$ is $a_0 + a_1 y + a_2 y^2$ where

$$a_0 = 2x^{312} + x^{311} + 2x^{310} + x^{308} + 4x^{306} + 2x^{305} + x^{304} + 3x^{303} + 2x^{302} + 3x^{301} + 3x^{300} + 3x^{297} + 4x^{295} + x^{294} + 3x^{292} + x^{291} + 3x^{290} + 2x^{289} + 2x^{288} + 4x^{287} + x^{286} + 3x^{285} + 4x^{284} + 4x^{283} + 3x^{282} + x^{281} + 3x^{279} + 2x^{278} + 3x^{277} + 2x^{276} + 4x^{274} + x^{272} + 4x^{271} + 2x^{270} + 2x^{269} + 2x^{267} + 4x^{266} + 4x^{265} + 4x^{264} + x^{263} + 2x^{262} + x^{260} + x^{257} + 3x^{256} + 3x^{255} + 2x^{254} + 4x^{253} + 2x^{252} + 3x^{251} + 3x^{250} + x^{248} + 4x^{247} + 2x^{246} + 3x^{245} + 2x^{243} + 2x^{242} + 4x^{241} + 4x^{240} + 4x^{237} + x^{234} + 3x^{233} + 2x^{232} + x^{231} + 3x^{228} + x^{227} + 4x^{226} + 3x^{224} + 3x^{223} + 2x^{222} + 4x^{221} + x^{219} + 3x^{217} + 3x^{216} + x^{215} + 3x^{214} + 3x^{210} + 2x^{209} + x^{207} + 4x^{205} + 4x^{203} + 3x^{202} + x^{201} + 3x^{199} + x^{198} + 2x^{197} + 4x^{196} + x^{195} + 4x^{194} + 2x^{193} + 3x^{192} + 4x^{191} + 4x^{190} + x^{189} + 2x^{188} + 2x^{186} + 4x^{185} + 4x^{184} + 2x^{181} + x^{180} + x^{179} + 3x^{178} + 2x^{177} + 2x^{176} + 4x^{173} + 3x^{172} + x^{171} + 4x^{170} + x^{169} + 3x^{168} + 4x^{167} + 3x^{166} + 4x^{161} + 2x^{159} + x^{158} + 4x^{157} + x^{155} + 3x^{154} + 4x^{153} + 2x^{152} + x^{151} + x^{150} + 2x^{149} + 2x^{148} + 2x^{147} + 2x^{146} + 4x^{145} + 4x^{143} + 2x^{141} + 2x^{139} + x^{138} + 4x^{137} + 4x^{135} + 3x^{133} + 3x^{132} + 2x^{131} + 4x^{129} + 3x^{128} + 4x^{127} + 2x^{126} + 4x^{125} + x^{122} + 4x^{121} + 3x^{120} + 3x^{119} + 2x^{118} + 3x^{115} + 4x^{114} + x^{113} + x^{110} + x^{109} + 4x^{108} + 3x^{107} + 2x^{106} + x^{105} + x^{104} + 2x^{103} + 3x^{100} + 4x^{99} + 2x^{98} + 3x^{97} + 2x^{96} + 4x^{95} + x^{94} + 2x^{92} + 2x^{91} + x^{90} + 4x^{89} + 3x^{86} + 3x^{85} + 4x^{83} + 2x^{82} + 4x^{81} + 2x^{80} + 3x^{79} + 2x^{77} + 2x^{76} + x^{75} + 3x^{74} + 2x^{73} + x^{71} + 2x^{70} + x^{69} + 4x^{68} + 4x^{67} + x^{66} + x^{64} + x^{62} + 3x^{61} + 2x^{59} + 3x^{58} + 4x^{56} + x^{55} + x^{53} + 2x^{52} + 4x^{51} + x^{49} + 4x^{47} + x^{46} + 3x^{45} + 4x^{44} + x^{43} + 4x^{42} + 3x^{41} + x^{40} + 4x^{39} + 2x^{37} + x^{36} + x^{34} + 3x^{33} + x^{31} + x^{30} + 4x^{29} + x^{27} + 3x^{26} + x^{25} + 3x^{24} + x^{23} + x^{22} + 3x^{21} + 4x^{19} + 2x^{18} + 4x^{17} + x^{16} + x^{15} + 4x^{14} + 2x^{13} + 4x^{12} + x^{11} + 3x^{10} + 2x^9 + 2x^8 + 2x^7 + 4x^6 + 2x^5 + 2x^4 + 4x^3 + 2x^2 + 2x + 4,$$

$$a_1 = x^{317} + 4x^{316} + 4x^{315} + x^{314} + 3x^{313} + 4x^{312} + 4x^{310} + 2x^{309} + 2x^{308} + x^{307} + x^{306} + 3x^{305} + 2x^{304} + x^{302} + x^{301} + 3x^{300} + 2x^{299} + 4x^{298} + 2x^{297} + 2x^{295} + 2x^{294} + 3x^{293} + 4x^{292} + x^{291} + 3x^{290} + 3x^{289} + 4x^{288} + 3x^{287} + x^{286} + 4x^{283} + 3x^{282} + 4x^{280} + 2x^{276} + x^{275} + 3x^{274} + 3x^{273} + 2x^{272} + x^{271} + 4x^{270} + x^{269} + 4x^{268} + 2x^{267} + 3x^{266} + x^{264} + 3x^{263} + 4x^{262} + 4x^{261} + 2x^{260} + 3x^{259} + x^{258} + x^{257} + x^{256} + 4x^{255} + x^{254} + 3x^{253} + 2x^{250} + 4x^{249} + 4x^{248} + 4x^{247} + 3x^{245} + 4x^{244} + x^{243} + 2x^{242} + x^{241} + 4x^{240} + x^{238} + x^{237} + 3x^{236} + 3x^{235} + x^{233} + 4x^{232} + 3x^{231} + 3x^{228} + 3x^{227} + x^{226} + 4x^{225} + x^{224} + 4x^{223} + 4x^{222} + 3x^{221} + 2x^{220} + 2x^{219} + 4x^{218} + 2x^{217} + x^{216} + 2x^{214} + 4x^{213} + 4x^{212} + 2x^{211} + 3x^{210} + x^{209} + 4x^{208} + 3x^{205} + x^{202} + 2x^{201} + 4x^{198} + 2x^{197} + 4x^{196} + 4x^{195} + 2x^{194} + x^{193} + 4x^{192} + 3x^{190} + 3x^{189} + 2x^{188} + 2x^{187} + 2x^{186} + 4x^{184} + 4x^{183} + 3x^{182} + x^{181} + 4x^{180} +$$

$$x^{179}+4x^{178}+x^{177}+x^{176}+3x^{174}+x^{173}+3x^{171}+4x^{170}+2x^{169}+2x^{166}+4x^{165}+x^{163}+4x^{162}+$$
$$4x^{161}+2x^{160}+4x^{159}+x^{158}+3x^{157}+4x^{156}+2x^{155}+2x^{154}+3x^{153}+2x^{152}+3x^{150}+4x^{148}+$$
$$3x^{147}+x^{146}+3x^{145}+3x^{144}+4x^{143}+x^{142}+4x^{141}+3x^{140}+2x^{139}+x^{138}+x^{136}+x^{135}+2x^{134}+$$
$$x^{133}+4x^{132}+x^{131}+x^{130}+x^{127}+3x^{126}+3x^{124}+x^{122}+4x^{120}+x^{119}+2x^{118}+4x^{117}+2x^{114}+$$
$$4x^{113}+2x^{111}+x^{110}+3x^{109}+3x^{108}+3x^{105}+x^{103}+x^{100}+3x^{97}+3x^{96}+3x^{95}+4x^{94}+4x^{93}+$$
$$4x^{92}+2x^{91}+3x^{90}+3x^{89}+x^{88}+2x^{87}+2x^{86}+2x^{85}+4x^{84}+x^{83}+4x^{81}+x^{80}+x^{79}+4x^{78}+2x^{77}+$$
$$3x^{76}+x^{75}+3x^{74}+3x^{73}+2x^{72}+x^{69}+4x^{68}+4x^{67}+4x^{66}+x^{65}+3x^{64}+4x^{63}+2x^{62}+4x^{61}+$$
$$2x^{60}+4x^{59}+3x^{58}+3x^{57}+4x^{56}+x^{55}+4x^{54}+x^{53}+3x^{51}+3x^{50}+4x^{48}+x^{46}+3x^{45}+4x^{44}+x^{42}+$$
$$3x^{40}+4x^{39}+2x^{38}+x^{37}+3x^{36}+2x^{35}+4x^{33}+4x^{32}+3x^{30}+x^{29}+4x^{26}+3x^{25}+x^{22}+2x^{21}+$$
$$3x^{19}+4x^{18}+2x^{17}+x^{16}+x^{15}+4x^{14}+x^{12}+3x^{11}+2x^{10}+4x^{9}+x^{8}+4x^{7}+2x^{5}+x^{3}+3x+4,$$

$$a_2 = x^{315}+2x^{314}+x^{313}+x^{311}+3x^{309}+3x^{308}+2x^{307}+2x^{306}+4x^{305}+2x^{304}+3x^{302}+4x^{301}+2x^{299}+$$
$$3x^{298}+3x^{296}+2x^{295}+3x^{294}+2x^{292}+3x^{291}+x^{290}+2x^{289}+4x^{288}+3x^{287}+4x^{286}+4x^{285}+$$
$$4x^{283}+x^{282}+4x^{281}+4x^{279}+4x^{278}+4x^{276}+2x^{275}+3x^{274}+x^{273}+3x^{272}+4x^{271}+3x^{268}+$$
$$4x^{267}+3x^{265}+x^{264}+4x^{263}+x^{260}+3x^{259}+3x^{258}+2x^{256}+2x^{255}+x^{254}+x^{253}+2x^{252}+4x^{250}+$$
$$4x^{249}+x^{248}+2x^{247}+2x^{246}+x^{245}+2x^{244}+3x^{243}+4x^{242}+2x^{241}+x^{240}+x^{236}+3x^{235}+4x^{233}+$$
$$3x^{232}+x^{231}+3x^{228}+x^{227}+3x^{226}+2x^{225}+3x^{224}+3x^{223}+2x^{222}+3x^{221}+x^{220}+4x^{219}+2x^{217}+$$
$$3x^{215}+4x^{213}+x^{212}+x^{211}+x^{210}+x^{209}+4x^{207}+x^{206}+2x^{204}+3x^{202}+2x^{201}+2x^{199}+3x^{197}+$$
$$x^{196}+4x^{195}+x^{194}+3x^{193}+2x^{192}+4x^{191}+x^{190}+2x^{189}+x^{187}+2x^{186}+2x^{184}+4x^{183}+3x^{182}+$$
$$4x^{180}+3x^{177}+4x^{175}+x^{174}+3x^{172}+4x^{170}+4x^{169}+3x^{167}+x^{166}+3x^{165}+3x^{164}+2x^{163}+$$
$$4x^{161}+4x^{160}+4x^{159}+4x^{158}+x^{157}+4x^{156}+2x^{155}+x^{154}+4x^{152}+x^{151}+3x^{149}+3x^{147}+2x^{146}+$$
$$2x^{145}+2x^{144}+x^{143}+3x^{142}+4x^{141}+4x^{140}+2x^{139}+4x^{138}+x^{137}+2x^{134}+3x^{133}+x^{132}+3x^{130}+$$
$$2x^{129}+4x^{127}+2x^{126}+x^{124}+2x^{123}+4x^{121}+2x^{119}+2x^{118}+4x^{115}+3x^{114}+x^{113}+4x^{111}+$$
$$2x^{109}+4x^{108}+3x^{106}+4x^{105}+4x^{104}+x^{103}+x^{102}+x^{100}+x^{99}+x^{98}+2x^{97}+x^{96}+x^{95}+x^{94}+$$
$$x^{93}+2x^{92}+4x^{91}+x^{88}+3x^{87}+4x^{86}+2x^{84}+x^{83}+3x^{82}+2x^{81}+2x^{80}+x^{79}+4x^{78}+x^{77}+x^{76}+$$
$$x^{75}+x^{74}+3x^{73}+2x^{71}+3x^{70}+x^{69}+3x^{68}+x^{66}+2x^{65}+3x^{64}+2x^{62}+x^{60}+x^{59}+x^{58}+3x^{56}+$$
$$2x^{55}+2x^{54}+4x^{53}+2x^{52}+2x^{51}+x^{49}+3x^{48}+2x^{47}+3x^{45}+3x^{44}+4x^{43}+2x^{41}+x^{40}+x^{39}+$$
$$3x^{38}+3x^{37}+2x^{36}+3x^{34}+x^{33}+3x^{32}+2x^{31}+4x^{30}+2x^{29}+3x^{27}+3x^{26}+4x^{25}+3x^{23}+3x^{22}+$$

$$4x^{20}+3x^{19}+2x^{17}+x^{15}+x^{14}+x^{13}+2x^{12}+4x^{10}+2x^9+3x^8+3x^7+2x^6+3x^5+3x^3+2x^2+x.$$

The fundamental unit ε has the values at the infinite places $v_{P_{1,\infty}}(\varepsilon) = -321$, $v_{P_{2,\infty}}(\varepsilon) = 367$, $v_{P_{3,\infty}}(\varepsilon) = -46$. The fundamental unit ε can be written in a compact representation $\mathbf{t} = (\mu, (\beta_1, \beta_2, \cdots, \beta_{10}))$ where $\mu, \beta \in O_F$, $\beta_i \in \frac{1}{\prod_{j=1}^{i-1} \beta_j^{2^{i-j}}} O_F$ for $1 \leq i \leq l$. μ and β_i computed using Algorithm 4 are as below.

$\mu = 2$, $\beta_1 = 2y$,

$\beta_2 = \beta_3 = 2y^2 + (2x^3 + 2)y + 2x^5 + 2x^4 + 2$,

$\beta_4 = \frac{3x^5+4x^4+3x^3+x^2+4x+2}{x^3+2x+1}y^2 + \frac{3x^8+4x^7+3x^6+4x^5+3x^4+x^2+4x+4}{x^3+2x+1}y +$
$\frac{3x^{10}+2x^9+2x^8+4x^7+4x^5+x^4+2x^3+2x^2+3x+3}{x^3+2x+1}$,

$\beta_5 = \frac{2x^2+1}{x^5+2x^4+4x^3+3x^2+2x}y^2 + \frac{2x^5+2x^3+x^2+2x+2}{x^5+2x^4+4x^3+3x^2+2x}y + \frac{2x^7+3x^6+4x^5+2x^4+x^3+3x^2+x+3}{x^5+2x^4+4x^3+3x^2+2x}$,

$\beta_6 = \frac{2x^4+3x^3+x+1}{x^4+4x^3+4x+2}y^2 + \frac{2x^7+3x^6+3x^4+4x^3+4x+2}{x^4+4x^3+4x+2}y + \frac{2x^9+3x^7+x^6+2x^5+3x^4+2x^3+x^2+4x+2}{x^4+4x^3+4x+2}$,

$\beta_7 = \frac{2x^6+2x^5+x^4+4x^3+x^2+3}{x^4+3x^3+3x^2+3x+4}y^2 + \frac{2x^9+2x^8+x^7+x^6+3x^5+x^4+2x^3+x^2+3x}{x^4+3x^3+3x^2+3x+4}y + \frac{2x^{11}+4x^{10}+3x^9+3x^6+4x^4+4x^2+2}{x^4+3x^3+3x^2+3x+4}$,

$\beta_8 = \frac{3x^5+2x^4+2x^3+2x^2+4x+1}{x^6+4x^5+2x^4+3x^3+2x^2+x+4}y^2 + \frac{3x^8+2x^7+2x^6+x^4+2x^3+x+1}{x^6+4x^5+2x^4+3x^3+2x^2+x+4}y + \frac{(3x^{10}+4x^8+4x^7+4x^6+x^5+x^4+x^3+3}{x^6+4x^5+2x^4+3x^3+2x^2+x+4}$,

$\beta_9 = \frac{2x^6+4x^5+x^4+3x^3+3x^2+3x+3}{x^5+3x^4+2x^3+4x^2+x}y^2 + \frac{2x^9+4x^8+x^7+2x^5+4x^4+4x^3+3x^2+4x+1}{x^5+3x^4+2x^3+4x^2+x}y +$
$\frac{2x^{11}+x^{10}+4x^8+x^7+3x^6+4x^5+x^4+3x^3+2x+4}{x^5+3x^4+2x^3+4x^2+x}$,

$\beta_{10} = (4x^5 + 4x^4 + 3x^3 + 4x^2 + 4)y^2 + (4x^8 + 4x^7 + 3x^6 + 3x^5 + 4x^4 + 2x^3 + 4x^2 + 1)y +$
$4x^{10} + 3x^9 + 2x^8 + 2x^7 + 4x^6 + 3x^5 + 3x^4 + 4x^2$.