KTH Numerical Analysis
and Computer Science

Department of Numerical Analysis and Computer Science

On the l-Ary GCD-Algorithm and Computing Residue Symbols

Douglas Wikström

TRITA-NA-0439

NADA is a co-operating departement between the
Royal Institute of Technology and Stockholm University

Douglas Wikström
On the l-Ary GCD-Algorithm and Computing Residue Symbols

Report number: TRITA-NA-0439
Publication date: November 15, 2004
E-mail of author: dog@nada.kth.se

Reports can be ordered from:

Numerical Analysis and Computer Science (NADA)
Royal Institute of Technology (KTH)
SE-100 44 Stockholm
SWEDEN

telefax: +46 8 790 09 30
http://www.nada.kth.se/

On the l-Ary GCD-Algorithm and Computing Residue Symbols

Douglas Wikström

November 15, 2004

Abstract

We present an l-ary GCD algorithm for the ring $\mathbb{Z}[\zeta]$, where ζ is a primitive eighth root of unity ζ. A corresponding algorithm for computing the octic residue symbol is also described. Both algorithms run in time $O(n^2)$ in the bitsize n of the input using naive arithmetic in $\mathbb{Z}$. As far as we know, this is the first report of a non-Euclidean gcd algorithm for the ring of integer for a non-quadratic number field, and the first octic residue symbol algorithm which runs in time $O(n^2)$.

1 Introduction

Given a and b, the greatest common divisor is the largest integer d such that $d \mid a$ and $d \mid b$. The corresponding problem can be considered for two elements α and β in any ring R with unique factorization. The problem of finding the greatest common divisor of two integers is one of the oldest problems studied in number theory. A precise understanding of the complexity of different gcd algorithms gives a better understanding of the arithmetic in the domain under consideration. Thus, our main motivation for studying this problem is its mathematical beauty.

The problem of quadratic residuosity can be described as follows. Given a prime p and an integer a determine if there exists a solution to the equation $x^2 = a \pmod{p}$, i.e. determine if a is a square in $\mathbb{Z}_p^*$. The Legendre symbol $(\frac{a}{p})$ is defined to be equal to 0 if $p \mid a$, equal to 1 if there is a solution to the equation and equal to -1 if there is no solution to the equation. For a non-prime b with decomposition $b = p_1 \cdot \ldots \cdot p_s$ into primes the Jacobi symbol $(\frac{a}{b})$ is defined in terms of the Legendre symbol by $(\frac{a}{b}) = \prod_{j=1}^{s} (\frac{a}{p_i})$. In contrast to the Legendre symbol, the Jacobi symbol may equal one without there being a solution to $x^2 = a \pmod{b}$. More generally, for two elements α and β in a ring R containing the mth roots of

unity, one can define an mth power residue symbol $\left(\frac{\alpha}{\beta}\right)_m$ with similar properties as the Jacobi symbol. A practical cryptographic application of a fast algorithm for computing the lth power residue symbol is to check membership in a group $G_q \subset (\mathbb{Z}/p\mathbb{Z})^*$ for prime numbers p and q such that $p = lq+1$ with $l < q$. Suppose that p factors into primes $p = \prod_{j=1}^{k} \pi_j$ in R. Then $a \in (\mathbb{Z}/p\mathbb{Z})^*$ is contained in G_q if and only if $\left(\frac{a}{\pi_1}\right)_l = 1$. This was our original motivation for studying this problem [12]. Although the algorithm we present has too large constant factors to be of immediate practical value, we hope that it is a first step in the program of finding an efficient practical algorithm.

Algorithmically, the problem of computing residue symbols is intimately connected to the problem of computing the gcd of elements, and there exists efficient algorithms for solving both problems in the integers and in some larger domains. It is an intriguing research problem to find efficient algorithms for as general domains as possible. The relation between the two problems make it natural to study them together.

1.1 Previous Work

The Euclidean gcd algorithm is well known. Less known are the alternative algorithms. Stein [8] introduced the binary gcd algorithm, which is particularly well suited for implementation on computers. The binary algorithm is based on the following facts

$$\begin{aligned} \gcd(a,b) &= 2\gcd(a/2, b/2) && \text{if } a \text{ and } b \text{ are even} \\ \gcd(a,b) &= \gcd(a, b/2) && \text{if } a \text{ is odd and } b \text{ is even} \\ \gcd(a,b) &= \gcd(a, (a-b)/2) && \text{if } a \text{ and } b \text{ are odd} \ . \end{aligned}$$

One may always apply one of the rules to reduce the size of elements, while preserving the gcd. Thus, by simply shifting and subtracting integers the gcd of two integers can be computed quickly.

Weilert [11] generalized this algorithm to the Gaussian integers. Damgård and Skjovbjerg Frandsen [2] independently also generalized the binary algorithm to the Eisenstein integers and the Gaussian integers.

Interestingly, the binary gcd algorithm can be "translated" to compute the Jacobi symbol. This was first described by Shallit and Sorenson [6]. Weilert [11] generalized this algorithm to the Gaussian integers. Independently, both Damgård and Skjovbjerg Frandsen [2] and Wikström [12] generalized the binary algorithm to the Eisenstein integers and the Gaussian integers in slightly different ways.

Kaltofen and Rolletschek [4] devised a gcd algorithm for the ring of integers in any quadratic number field. Their approach is based on the idea of given elements

α and β to find an integer j such that the norm of $j\alpha \bmod \beta$ is smaller than the norm of β. It turns out that this is always possible with $|j|$ bounded essentially by the square root of the discriminant.

Sorenson [7] introduced the l-ary algorithm, which generalizes the binary algorithm. The l-ary algorithm is based on the result by Minkowski that given a and b one can find c and d such that $ca + db = 0 \bmod l$ for an integer l, where $|a|, |b| \leq \sqrt{l}$. This is an analog to the binary algorithm, in that in each iteration the size of the largest integer is reduced roughly by a factor $2\sqrt{l}/l$. The details of this algorithm is however more involved than the binary algorithm. Sorenson also constructed a parallel version of his algorithm. Weilert [10] generalized also this algorithm to the Gaussian integers.

Recently Agarwal and Skovbjerg Frandsen [1] gave an algorithm related to both the binary and the l-ary algorithms for computing gcd in several complex quadratic rings. Their result is intriguing in that they compute gcd in the ring of integers $D = \mathbb{Z}[\frac{1}{2}(-1 + \sqrt{-19})]$ of $\mathbb{Q}(\sqrt{-19})$ which is not norm-Euclidean, i.e. although there exists a Euclidean algorithm in D the height function is not the norm.

1.2 Contribution

We extend Sorenson's [7] l-ary algorithm to compute the gcd of α and β in $\mathbb{Z}[\zeta_8]$, where $\zeta = \zeta_8$ is a primitive eighth root of unity. Our algorithm runs in time $O(n^2)$. We also construct a corresponding algorithm for computing the octic residue symbol of α modulo β, with the same running time.

As far as we know, we give the first non-Euclidean gcd algorithm in the ring of integers of a non-quadratic field, and the first algorithm for computing the octic residue symbol in time $O(n^2)$ using naive arithmetic for $\mathbb{Z}$. This should be compared to the straightforward computation of the residue symbol which takes time $O(n^3)$ in the same model.

Thus, we present some results in the program to find non-Euclidean algorithms for computing the gcd and power residue symbols in as general settings as possible. The octic case is interesting, since it introduces some, but not all, difficulties found in the general settings.

1.3 Outline of Paper

First we introduce some basic notation. Then we recall some well known facts about the octic cyclotomic field $\mathbb{Q}(\zeta)$ and its ring of integers D. We continue by presenting the octic residue symbol and the reciprocity laws it satisfies. Our main references for this part of the paper are Ireland and Rosen [3] and Chapter 9 in Lemmermeyer [9].

We proceed by proving a number of results for $\mathbb{Q}(\zeta)$ and D, on which our algorithms and the analysis of the algorithms are based. This part of the paper is divided into a number of subsections. First we prove basic facts about the primes and primary elements in D. Then we define the notion of a balanced element and establish some properties of such elements. We then prove a weak triangle law for the ring of integers in any number field. In the three subsections that follow we consider the problem of given α and β to find elements γ and δ such that $\gamma\alpha + \delta\beta$ is divisible by l and has relatively small norm. That is, we establish the results that allow us to generalize Sorenson's [7] l-ary algorithm.

Similarly to the Euclidean algorithm the $l$ary algorithm must in each step find the largest of its inputs, and if necessary swap them. Normally the "size" of an element is measured by its norm, but the norm is to expensive to compute in each iteration. Thus, we consider the problem of finding the "size" of elements without computing their norm. We do this by first introducing the notion of a balanced element, i.e. an element such that the complex absolute value of all its conjugates are roughly equal. Then we give a function which upper bounds the norm, and approximates the norm well for balanced elements (it does not approximate the norm for general elements). Then we show that this measure can be efficiently approximated. The idea of balancing elements appears implicitly in the work of Kaltofen and Rolletschek [4].

Finally, we describe the algorithms in detail and then prove that they are correct and that their running time is $O(n^2)$.

2 Notation

We write $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ for the rational integers, the rational numbers, the real numbers, and the complex numbers. The imaginary unit is denoted by $i = \sqrt{-1}$. We denote the complex absolute value by $|\cdot| : \mathbb{C} \to \mathbb{R}$, where $|a + bi| = \sqrt{a^2 + b^2}$.

3 Naive Bit Complexity

We follow Sorenson [7] and analyze our algorithms in the naive bit complexity model. This means that we assign the following costs for integer arithmetic where x and y are integers. Without loss we consider only positive integers.

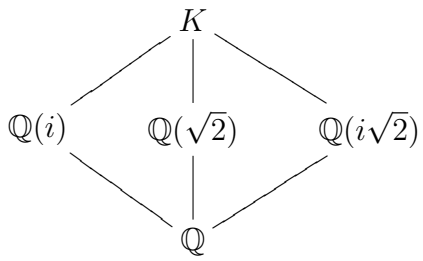
- Copying the value of x takes time $O(\log x)$.
- Computing $x + y$ or $x - y$ takes time $O(\log x + \log y)$.
- Computing xy, x/y, or $x \bmod y$ takes time $O(\log x \log y)$.

- Comparing x with y takes time $O(\log x + \log y))$.

4 The Octic Cyclotomic Field and its Ring of Integers

In this section we introduce some notation and recall results on the eighth cyclotomic field and its ring of integers.

Write $\zeta = e^{2\pi/8} = \frac{1}{2}(\sqrt{2} + \sqrt{2}i)$. Then ζ is a primitive eighth root of unity. We consider the cyclotomic field K formed as $K = \mathbb{Q}(\zeta)$, and write D for the ring of integers in K. The field K is a biquadratic number field, i.e. it can be formed by two extensions of degree 2, $K = \mathbb{Q}(i, \sqrt{2})$. The lattice of subfields of K is illustrated below.



Furthermore, the integers D is a principal ideal domain, and thus have unique factorization, since the class number of K is one (cf. Theorem 11.1 in [9]), and $D = \mathbb{Z}[\zeta] = \mathbb{Z} + \mathbb{Z}\zeta + \mathbb{Z}\zeta^2 + \mathbb{Z}\zeta^3$.

The relative norm of an extension K/L of an element α in K is defined by the product $N_{K/L}\alpha = \prod_{\sigma \in \mathrm{Gal}(K/L)} \alpha^\sigma$, where $\sigma : \alpha \mapsto \alpha^\sigma$ are the isomorphisms of the Galois group $\mathrm{Gal}(K/L)$ of the extension K/L. We write N to denote the norm $N_{K/\mathbb{Q}}$. For $\alpha \in D$ we have $N\alpha \in \mathbb{Z}$. By abuse of notation we write $N\mathfrak{a} = |D/\mathfrak{a}|$ to denote the norm of an ideal $\mathfrak{a} \in D$, since D is a principal ideal domain.

We say that an element $\gamma \in D$ is a unit if there exists an element ω such that $\gamma\omega = 1$, this is equivalent with $N\gamma = 1$. Two elements α and β in D are said to be associates if there exists a unit $\gamma \in D$ such that $\alpha = \gamma\beta$. The units of D are generated as a multiplicative group by ζ and $\epsilon = 1 + \sqrt{2}$, i.e. any unit γ can be written $\gamma = \zeta^{j'}\epsilon^j$ for $j' \in [0, 7]$ and $j \in \mathbb{Z}$.

An element $\alpha \in D$ is said to divide $\beta \in D$ if there exists a $\gamma \in D$ such that $\beta = \alpha\gamma$, and this is written $\alpha \mid \beta$. The greatest common divisor $\gamma = \gcd(\alpha, \beta)$ of α and β is only defined up to multiplication by a unit. The element γ is characterized by $\gamma \mid \alpha$ and $\gamma \mid \beta$ and that $\gamma' \mid \alpha$ and $\gamma' \mid \beta$ implies that $\gamma' \mid \gamma$.

An element $\alpha \in D$ is said to be prime if $\alpha \mid \beta\gamma$, $\beta, \gamma \in D$ implies that $\alpha \mid \beta$ or $\alpha \mid \gamma$. An element $\alpha \in D$ is called irreducible if it has the property that $\alpha = \beta\gamma$

implies that either $\beta \in D$ or $\gamma \in D$ is a unit. For principal ideal domains like D the two notions coincide. Two elements α and β in D are said to be relatively prime if the ideal $(\alpha, \beta) = D$, or equivalently $\gcd(\alpha, \beta)$ is a unit.

5 The Octic Residue Symbol

In the previous section we recalled general facts about the field K and its ring of integers D. In this section we review more specialized definitions and results about the octic residue symbol.

Definition 1 (Primary). An element $\alpha \in D$ is *primary* if $\alpha = 1 \bmod (2 + 2\zeta)$.

The notion of a primary element can be generalized to larger domains. However, there seems to be no consensus on a general definition. Our definition is taken from Chapter 9 of [5].

Definition 2 (Octic Residue Symbol). Let α and β in D and let β be a non-unit with $N\beta = 1 \pmod 8$.

1. If α and β are not relatively prime we define $\left(\frac{\alpha}{\beta}\right) = 0$.
2. If α and β are relatively prime then
   (a) if β is prime then define $\left(\frac{\alpha}{\beta}\right)$ to be the unique 8th root of unity such that
   $$\left(\frac{\alpha}{\beta}\right) = \alpha^{\frac{N\beta-1}{8}} \pmod \beta \ .$$
   (b) if β has prime factorization $\beta = \prod_{j=1}^{s} \beta_j^{r_j}$, then define
   $$\left(\frac{\alpha}{\beta}\right) = \prod_{j=1}^{s} \left(\frac{\alpha}{\beta_j}\right)^{r_j} \ .$$

Proposition 3. *Let α and β in D be relatively prime to γ in D and let γ be primary.*

1. *Then we have*
   $$\left(\frac{\alpha\beta}{\gamma}\right) = \left(\frac{\alpha}{\gamma}\right)\left(\frac{\beta}{\gamma}\right) \ .$$

2. If furthermore $\alpha = \beta \bmod \gamma$*, then we also have*

$$\left(\frac{\alpha}{\gamma}\right) = \left(\frac{\beta}{\gamma}\right) \ .$$

Given an $\alpha \in D$ we can write it in different basis, $\alpha = a(\alpha) + b(\alpha)i$, $\alpha = c(\alpha) + d(\alpha)\sqrt{-2}$, and $\alpha = e(\alpha) + f(\alpha)\sqrt{2}$. These expressions correspond to the different towers of fields in the lattice of subfields of K, and are used in the statement of the reciprocity laws below.

Theorem 4 (Octic Reciprocity Law). *If* α *and* β *are relatively prime and primary non-units, then*

$$\left(\frac{\alpha}{\beta}\right) = (-1)^{\frac{N\alpha-1}{8}\frac{N\beta-1}{8}} \zeta^{d(\alpha)f(\beta)-d(\beta)f(\alpha)} \left(\frac{\beta}{\alpha}\right) \ .$$

Theorem 5 (Supplementary Octic Reciprocity Laws). *Let* $\alpha \in D$ *be a primary non-unit. Then*

$$\begin{aligned}
\left(\frac{1-\zeta}{\alpha}\right) &= \zeta^{\frac{5a-5+5b+18d+b^2-2bd+d^4/2}{8}} \\
\left(\frac{1+\zeta}{\alpha}\right) &= \zeta^{\frac{a-1+b+6d+b^2+2bd+d^4/2}{8}} \\
\left(\frac{\zeta}{\alpha}\right) &= \zeta^{\frac{a-1+4b+2bd+2d^2}{4}} \\
\left(\frac{1+\zeta+\zeta^2}{\alpha}\right) &= \zeta^{\frac{a-1-2b+2d-2d^2}{4}} \\
\left(\frac{\epsilon}{\alpha}\right) &= \zeta^{\frac{d-3b-bd-2d^2}{2}} \ .
\end{aligned}$$

6 Results On Rings of Integers in Number Fields

This section develops results that allow us to construct and analyze the gcd algorithm and the octic residue symbol algorithm in D.

6.1 Primes and Primary Elements

Lemma 6. *The prime factorization of* 2 *is given by*

$$2 = (1+\zeta)(1+\zeta^3)(1+\zeta^5)(1+\zeta^7) = -(1-\zeta)^2(1+\zeta)^2 \ .$$

Proof. Equality follows by calculation. Let $1+\zeta = \alpha\beta$ and take the norm on both sides. Since $N(1+\zeta) = N\alpha N\beta = 2$, this implies that either α or β must be a unit, so $1+\zeta$ is prime. The other factors on the right are prime for the same reason. The second equality follows from the fact that $1+\zeta^5 = \zeta^5(1+\zeta^3)$, $1+\zeta^3 = 1-\zeta$, $1+\zeta^7 = \zeta^7(1+\zeta)$, and $\zeta^5\zeta^7 = \zeta^4 = -1$. □

Lemma 7. *If $\alpha \in D$ has even norm $N\alpha$, then $(1+\zeta^j) \mid \alpha$ for some $j \in \{1,3,5,7\}$.*

Proof. Suppose that the prime factorization is given by $\alpha = \prod_{j=1}^{s} \gamma_j^{r_j}$ for $s \in \mathbb{N}$ and $r_j \in \mathbb{Z}$. Taking the norm on both sides and using the multiplicativity of the norm implies that $2 \mid N\gamma_j$ for some j, i.e. $N\gamma_j = (1+\zeta)(1+\zeta^3)(1+\zeta^5)(1+\zeta^7)\delta$ for some δ. Since γ_j is prime, δ must be a unit. From Lemma 6 we have $2 = (1+\zeta)(1+\zeta^3)(1+\zeta^5)(1+\zeta^7)$, and the lemma follows. □

Lemma 8. *If $\alpha \in D$ have odd norm $N\alpha$, then there exists $j,l \in \{0,1,2,3\}$ such that $\zeta^j\epsilon^l\alpha$ is primary.*

Proof. Let $\alpha = a_0 + a_1\zeta + a_2\zeta^2 + a_3\zeta^3$. First we argue that there is a unit γ such that $\gamma\alpha = b \bmod (2+2\zeta)$ for an odd $b \in \mathbb{Z}$. Consider the norm $N\alpha = 2a_1^2a_3^2 + a_2^4 - 4a_0a_1^2a_2 + a_1^4 + 4a_0a_3^2a_2 + a_0^4 + a_3^4 + 4a_1a_0^2a_3 - 4a_1a_2^2a_3 + 2a_2^2a_0^2$. We have $N\alpha = a_0^4 + a_1^4 + a_2^4 + a_3^4 \bmod 2$. Since the norm is odd by assumption, this implies that an odd number of the a_j for $j = 0,1,2,3$ are odd.

If a_j is odd but no other, we have that $\zeta^{4-j}\alpha = b_0 + b_1\zeta + b_2\zeta^2 + b_3\zeta^3$, where b_0 is odd and b_l for $l = 1,2,3$ are even. Thus, $\zeta^{4-j}\alpha = b_0 - b_1 + b_2 - b_3 \bmod (2+2\zeta)$, where we have used that $2\zeta = -2 \bmod (2+2\zeta)$. Thus, $\zeta^{4-j}\alpha = b \bmod (2+2\zeta)$ for an odd $b \in \mathbb{Z}$.

If a_j is even but no other, we have that $\zeta^{2-j}\alpha = b_0+b_1\zeta+b_2\zeta^2+b_3\zeta^3$, where b_2 is even. Thus, we have $\zeta^{2-j}\alpha = b_0'+b_1\zeta+b_3\zeta^3 \bmod (2+2\zeta)$, where $b_0' = b_0+b_2$ is odd. Furthermore, we have $\epsilon\zeta^{2-j}\alpha = (b_0'+b_1-b_3)+(b_0'+b_1)\zeta+(b_1+b_3)\zeta^2+(b_3-b_0')\zeta^3 = (b_0'+b_1-b_3)-(b_0'+b_1)+(b_1+b_3)-(b_3-b_0') = b_1-b_0'-b_3 \bmod (2+2\zeta)$, where $b_1-b_0'-b_3$ is odd.

Thus, we may define $\beta = \epsilon^b\zeta^j$, where $b \in \{0,1\}$ and $j \in \{0,1,2,3\}$ are chosen such that $\beta = b \bmod (2+2\zeta)$ for an odd $b \in \mathbb{Z}$. If $b = 1 \bmod 4$, we are done since by Lemma 6, $(2+2\zeta) = 2(1+\zeta) \mid 4$. Assume that $b = 3 \bmod 4$. It suffices to note that $\epsilon^2 = 3+2\zeta-2\zeta^3 = 3 \bmod (2+2\zeta)$, since then $\epsilon^2\beta = 9 = 1 \bmod (2+2\zeta)$. □

Lemma 9. *If $\alpha \in D$ is primary then $N\alpha = 1 \pmod 8$.*

Proof. Set $\alpha = 1+2(1+\zeta)\beta$, where $\beta = b_0 + b_1\zeta + b_2\zeta^2 + b_3\zeta^3$. Then $N\alpha = 1+8b_0-8b_3+128b_0^2b_1b_3+32b_3^4+32b_1b_2^2+32b_3^2b_2+32b_1b_2+32b_0^2b_2-64b_3b_1b_2-32b_3^2b_1-32b_1^2b_0-32b_0^2b_1-32b_1^2b_3-64b_0b_3b_2+64b_2b_0b_1+8b_2^2-32b_1^2b_2+32b_2^3+32b_0^4+64b_3b_1b_0-32b_3^3-128b_2^2b_3b_1+128b_2b_3^2b_0-128b_1^2b_2b_0+32b_0b_2^2+24b_0^2-32b_0b_3+24b_3^2+64b_3^2b_1^2+16b_0b_2+32b_2^4+32b_0^3+32b_1^4+16b_1b_3+64b_2^2b_0^2-32b_0^2b_3+32b_0b_3^2-32b_1^3+32b_3b_2^2+8b_1^2 = 1 \pmod 8$. The reader may wish to verify this using a computer algebra system. □

6.2 Balancing the Complex Absolute Value of Algebraic Conjugates

Consider the algebraic norm $N\alpha = \prod_{\sigma \in \mathrm{Gal}(K/\mathbb{Q})} \alpha^\sigma$ of an element $\alpha \in D$. Note that the complex absolute value of ζ equals that of ζ^7, and the complex absolute value of ζ^3 equals that of ζ^5. Thus, $N\alpha = \alpha\overline{\alpha}\alpha_3\overline{\alpha}_3 = |\alpha|^2 \cdot |\alpha_3|^2$, where $\alpha_3 = \alpha^{\sigma_3}$. This suggests that the complex absolute value of an element may give little information about its norm.

Recall that the group of units in D is generated by ζ and $\epsilon = 1+\sqrt{2} = 1+\zeta-\zeta^3$. Set $\epsilon_3 = \epsilon^{\sigma_3} = 1-\sqrt{2}$, then $N\epsilon = N\epsilon_3 = (\epsilon\epsilon_3)^2 = (-1)^2 = 1$ as expected. However, we have $\epsilon^{-1} = -\epsilon_3$, $|\epsilon|^2 = 3+2\sqrt{2} > 1$ and $|\epsilon^{-1}|^2 = 3-2\sqrt{2} < 1$. Thus, although $N(\epsilon^j) = 1^j = 1$, $|\epsilon^j|$ may be arbitrarily large for increasing j. This establishes what we suspected, that the complex absolute value says little about the norm.

Consider now an arbitrary element $\alpha \in D$. We have that $|\alpha| = |\alpha^{\sigma_7}|$ and $|\alpha_3| = |\alpha^{\sigma_5}|$, i.e. we may organize the conjugates of α into pairs having the same complex absolute value. Suppose that $|\alpha|$ is much smaller than $|\alpha^{\sigma_3}|$. Then we can multiply α by ϵ and get $N\alpha = N(\alpha\epsilon) = |\alpha\epsilon|^2|\alpha^{\sigma_3}\epsilon^{-1}|$. This implies that $|\alpha| < |\alpha\epsilon|$ and $|\alpha^{\sigma_3}| > |\alpha^{\sigma_3}\epsilon^{\sigma_3}|$. We can repeat this procedure until no longer possible, at which point we know that $|\alpha|$ and $|\alpha^{\sigma_3}|$ are of roughly equal size. If $|\alpha|$ is greater than $|\alpha^{\sigma_3}|$ we instead multiply α by ϵ^{-1}.

Informally, we could say that we can balance the complex absolute values of the algebraic conjugates of α. We introduce the following definition for a general number field K.

Definition 10 (Δ-Balanced Element). We say that a non-zero α in K is Δ-*balanced* if

$$\frac{|\alpha^\sigma|}{|\alpha^{\sigma'}|} \leq \Delta$$

for $\sigma, \sigma' \in \mathrm{Gal}(K/\mathbb{Q})$.

Note that α is Δ-balanced precisely when all of its conjugates are Δ-balanced, and that the requirement in the definition is equivalent to $\frac{1}{\Delta} \leq \frac{|\alpha^\sigma|}{|\alpha^{\sigma'}|}$.

For our favorite ring D we have the following lemma.

Lemma 11 (Balancing Lemma in D). *Let $\alpha \in D$ be non-zero. Then $\epsilon^j\alpha \in D$ is ϵ^2-balanced, where j is the integer closest to* $\log_{\epsilon^2} \frac{|\alpha^{\sigma_3}|}{|\alpha|}$.

Proof. Suppose that a non-zero $\alpha \in D$ is not ϵ^2-balanced. Then there are $\sigma, \sigma' \in \{\sigma_1, \sigma_3, \sigma_5, \sigma_7\}$ such that $\frac{|\alpha^\sigma|}{|\alpha^{\sigma'}|} > \epsilon^2$. We can not have $\sigma, \sigma' \in \{\sigma_1, \sigma_7\}$ or $\sigma, \sigma' \in \{\sigma_3, \sigma_5\}$, since then $\frac{|\alpha^\sigma|}{|\alpha^{\sigma'}|} = 1$. Recall also that $|\alpha_1| = |\alpha_7|$ and $|\alpha_3| = |\alpha_5|$, so it suffices to consider the fractions $\frac{|\alpha_3|}{|\alpha|}$ and $\frac{|\alpha_3|}{|\alpha|}$.

We have

$$\frac{|\alpha\epsilon^j|}{|\alpha^{\sigma_3}\epsilon^{-j}|} = (\epsilon_2)^j \frac{|\alpha|}{|\alpha^{\sigma_3}|} \ .$$

From the definition of j as the integer closest to $\log_{\epsilon^2} \frac{|\alpha^{\sigma_3}|}{|\alpha|}$ we have

$$\frac{1}{\epsilon^2} \leq \frac{|\alpha\epsilon^j|}{|\alpha^{\sigma_3}\epsilon^{-j}|} \leq \epsilon^2$$

from which the lemma follows. □

The lemma makes it natural to refer to ϵ^2-balanced elements simply as *balanced elements*.

Lemma 12. *The number of Δ-balanced units in D is bounded by $4\log_\epsilon \Delta$.*

Proof. Recall from Section 4 that every unit α is on the form $\zeta^{j'}\epsilon^j$ for $j' \in [0,7]$ and $j \in \mathbb{Z}$. Similarly to the proof above we need only consider the σ_3-conjugate. We have $\alpha_3 = \alpha^{\sigma_3} = (\zeta^{j'}\epsilon^j)^{\sigma_3} = \zeta^{3j'}\epsilon^{-j}(-1)^j$. Thus,

$$\frac{|\alpha|}{|\alpha_3|} = \epsilon^{2j} \ .$$

This implies that $|j|$ is bounded by $\frac{\log_\epsilon \Delta}{2}$. For each value of j there are at most 8 units and the claim follows. □

6.3 A Weak Triangle Inequality

It would be nice if given $\alpha, \beta \in D$, we had $N(\alpha+\beta) \leq c\max\{N\alpha, N\beta\}$ for a constant c, i.e. some type of "triangle inequality". Unfortunately, no such inequality exists in D. To see this it suffices to consider the example $N(\epsilon^j + \epsilon^{-j}) = |(\epsilon^j + \epsilon^{-j})|^4 \geq \epsilon^{4j}$, which gets arbitrarily large for increasing j.

Thus, there is no hope of finding a general triangle inequality. Instead we establish a triangle inequality for Δ-balanced elements.

Theorem 13 (Triangle Inequality for Δ-balanced Elements). *Let $\alpha, \beta \in D$ be Δ-balanced elements. Then we have*

$$N(\alpha+\beta) \leq 2^2(1+\Delta)^2 \max\{N\alpha, N\beta\} \ .$$

Moreover, if α and β are balanced we have $N(\alpha+\beta) \leq 187\max\{N\alpha, N\beta\}$.

Proof. Write $\alpha_3 = \alpha^{\sigma_3}$ and $\beta_3 = \beta^{\sigma_3}$. Note that

$$\begin{aligned} |\alpha\beta_3| + |\alpha\beta_3| &= |\alpha| \cdot |\beta_3| + |\alpha_3| \cdot |\beta| \leq 2\max\{|\alpha|, |\alpha_3|, |\beta|, |\beta_3|\}^2 \\ &\leq 2\Delta \max\{|\alpha\alpha_3|, |\beta\beta_3|\} \ , \end{aligned}$$

where we use that α and β are Δ-balanced, i.e. $\frac{1}{\Delta} \leq \frac{|\alpha|}{|\alpha_3|} \leq \Delta$. Then using the triangle inequality and multiplicativity of the complex absolute value, we have

$$\begin{aligned} N(\alpha+\beta) &= |(\alpha+\beta)(\alpha_3+\beta_3)|^2 = |\alpha\alpha_3 + \beta\beta_3 + \alpha\beta_3 + \alpha_3\beta|^2 \\ &\leq \left(|\alpha\alpha_3| + |\beta\beta_3| + |\alpha\beta_3| + |\alpha_3\beta|\right)^2 \\ &\leq \left(2(1+\Delta)\max\{|\alpha\alpha_3|, |\beta\beta_3|\}\right)^2 \\ &\leq 4(1+\Delta)^2 \max\{N\alpha, N\beta\} \ . \end{aligned}$$

□

Unfortunately, the inequality is not strong enough to generalize the previous binary-like algorithms [10, 2, 12] naively. If we could keep the elements balanced in each step we could use the triangle inequality. However, in each step the difference of two primary elements is only divisible by $2(1+i)$ and $N(2(1+i)) = 2^5 = 32$ which is much less than 187. Thus, even for balanced elements the triangle inequality is simply not strong enough. Even worse, to use the idea that the difference of two primary elements is divisible by $2(1+i)$, both elements must also be primary. This may force us to multiply the elements by ϵ, which would unbalance the elements.

Instead we extend the l-ary algorithm of Sorensen [7], which decreases the size in each iteration by a factor that we may choose to be large enough in relation to the constant of the triangle inequality.

6.4 A Certain Set of Elements

In this section we exhibit a relatively large set of elements with relatively small norm. These are not used directly as co-factors in the l-ary approach, but play an important role in the construction of such factors. Note that there obviously exists infinitely many distinct elements with small norm in D (and in most general settings) since $N(\epsilon^j) = 1$ for all $j \in \mathbb{Z}$, but we need the elements in our sets to have special properties.

Consider the set of elements

$$S_{l,t} = \{a_0 + a_1\zeta + a_2\zeta^2 + a_3\zeta^3 \mid 0 \leq a_i < l^{t/4}\} \backslash \{0\} \ .$$

We prove that the complex absolute value of any difference of two elements from $S_{l,t}$ is small and that the set $S_{l,t}$ contains relatively many elements. More precisely we have the following lemma.

Lemma 14. *Let $\gamma, \gamma' \in S_{l,t}$, where $l \geq 6$ and $t > 2$, and let $\gamma_3 = \gamma^{\sigma_3}$ and $\gamma'_3 = \gamma'^{\sigma_3}$. Then*

1. *$|\gamma - \gamma'|, |\gamma_3 - \gamma'_3| \leq 2\sqrt{\epsilon} \cdot l^{t/4}$, and*
2. *$|S_{l,t}| \geq \frac{l^t}{2}$.*

Proof. The square of the complex complex absolute value of an element $\alpha \in D$, where $\alpha = a_0 + a_1\zeta + a_2\zeta^2 + a_3\zeta^3$, is given by

$$|\alpha|^2 = \sum_{j=0}^{3} a_j^2 + \sqrt{2}(a_0a_1 - a_0a_3 + a_2a_3 + a_1a_2) \ .$$

Let $\gamma = c_0 + c_1\zeta + c_2\zeta^2 + c_3\zeta^3$ and let $\gamma' = c'_0 + c'_1\zeta + c'_2\zeta^2 + c'_3\zeta^3$. Then set $\alpha = \gamma - \gamma' = a_0 + a_1\zeta + a_2\zeta^2 + a_3\zeta^3$, where $a_j = c_j - c'_j$. Since both c_j and c'_j are positive, we have $|a_j| < l^{t/4}$. Taking the absolute value on both sides of Equation 1 and using the triangle inequality for the absolute value gives

$$\begin{aligned} |\alpha|^2 &\leq 4(1+\sqrt{2}) \max\{|a_0^2|, |a_1^2|, |a_2^2|, |a_3^2|, |a_0a_1|, |a_0a_3|, |a_2a_3|, |a_1a_2|\} \\ &\leq 4\epsilon l^{t/2} \ . \end{aligned}$$

Mutatis mutandi the argument shows the same thing for $\gamma_3 - \gamma'_3$.

There are at least $(l^{t/4} - 1)^4 - 1$ elements in $S_{l,t}$, since each coefficient can take at least $l^{t/4} - 1$ values and we subtract 1 for the 0. This expression is greater than $\frac{l^t}{2}$ for $l > l_0$ for some l_0. Numerical checking shows that $l_0 \leq 6$ for $t \geq 2$. □

6.5 Linear Combinations

In this section we construct the co-factors of the l-ary approach. Denote by $T_{l,t}$ the set of pairwise differences $T_{l,t} = \{\gamma - \gamma' \mid \gamma, \gamma' \in S_{l,t}\}$. We show that for any Δ-balanced elements $\alpha, \beta \in D$, we can find elements $\gamma, \delta \in T_{l,t}$ such that $l \mid (\gamma\alpha + \delta\beta)$ and still keep $N(\gamma\alpha + \delta\beta)$ relatively small. More precisely we have the following lemma.

Lemma 15. *Let $\alpha, \beta \in D$ be Δ-balanced and let $l \geq 6$ and $t = 2 + \frac{2}{\log_2 l}$. Then there exists $\gamma, \delta \in D$ such that*

1. *$l \mid (\gamma\alpha + \delta\beta)$, and*
2. *$N(\gamma\alpha + \delta\beta) \leq 64\epsilon^2(1+\Delta)^2 l^t \max\{N\alpha, N\beta\}$.*

Proof. Consider the map $f_{\alpha,\beta} : D^2 \to D/(l)$, $f_{\alpha,\beta} : (\gamma, \delta) \mapsto \gamma\alpha + \delta\beta + (l)$ for $\alpha, \beta \in D$. We have $|D/(l)| = N((l)) \geq N(l) = l^4$, and $|S_{l,t}^2| \geq \frac{l^{2t}}{4} > l^4$, i.e. the restriction of $f_{\alpha,\beta}$ to $S_{l,t}^2$ is not injective.

Thus, there exists $\gamma', \delta', \gamma'', \delta'' \in S_{l,t}$ such that $\gamma'\alpha + \delta'\beta = \gamma''\alpha + \delta''\beta$. We set $\gamma = \gamma' - \gamma''$ and $\delta = \delta' - \delta''$. Then from linearity of $f_{\alpha,\beta}$, (γ, δ) is in its kernel and $l \mid \gamma\alpha + \delta\beta$. This proves the first claim.

A weaker version of the second claim follows directly from the triangle inequality for Δ-balanced elements, but since this inequality greatly influences the running time of the algorithms we prove the tighter bound above directly.

Using the multiplicativity of the complex absolute value and Lemma 14 the second claim follows since

$$\begin{aligned}
N(\gamma\alpha + \delta\beta) &= |(\gamma\alpha + \delta\beta)(\gamma_3\alpha_3 + \delta_3\beta_3)|^2 \\
&= |\gamma\gamma_3\alpha\alpha_3 + \delta\delta_3\beta\beta_3 + \gamma\delta_3\alpha\beta_3 + \gamma_3\delta\alpha_3\beta|^2 \\
&\leq (|\gamma\gamma_3\alpha\alpha_3| + |\delta\delta_3\beta\beta_3| + |\gamma\delta_3\alpha\beta_3| + |\gamma_3\delta\alpha_3\beta|)^2 \\
&\leq (2\max\{|\gamma\gamma_3\alpha\alpha_3|, |\delta\delta_3\beta\beta_3|\} \\
&\quad +2\max\{|\gamma\delta_3|, |\gamma_3\delta|\}\max\{|\alpha|, |\alpha_3|, |\beta|, |\beta_3|\})^2 \\
&\leq 4(4\epsilon l^{t/2}\max\{|\alpha\alpha_3|, |\beta\beta_3|\} \\
&\quad +4\epsilon l^{t/2}\Delta\max\{|\alpha\alpha_3|, |\beta\beta_3|\})^2 \\
&= 64\epsilon^2 l^t(1+\Delta)^2\max\{N\alpha, N\beta\} \ .
\end{aligned}$$

□

Remark 1. The definition of t may seem overly complicated, i.e. one could use any constant $2 < t < 4$. But it minimizes the constant factor of our algorithms, and since these are relatively large we think that the more complicated definition is justified.

6.6 Spurious Factors

In each iteration of the algorithms, one of the inputs α and β is replaced by $\alpha, \gamma\alpha + \beta\gamma$ for $\gamma, \delta \in T_{l,t}$. Sorenson [7] notes that $\gcd(\alpha, \beta) = \gcd(\gamma\alpha + \beta\gamma, \beta)$ may not hold. Fortunately, the following lemma explains this completely.

Lemma 16. *Let $\alpha, \beta, \gamma, \delta \in D$. Then*

$$\gcd(\alpha, \beta) \mid \gcd(\gamma\alpha + \delta\beta, \beta) \quad , \quad \textit{and} \quad \frac{\gcd(\gamma\alpha + \delta\beta, \beta)}{\gcd(\alpha, \beta)} \;\Bigg|\; \gamma \ .$$

Proof. This follows from $\gcd(\gamma\alpha + \delta\beta, \beta) = \gcd(\gamma\alpha, \beta)$. □

6.7 Approximating the Norm of a Δ-Balanced Element

The norm of an element gives in some sense the "size" of the element. Unfortunately, the way the norm is defined requires computing a multiplication, which takes time $O(n^2)$ in the naive arithmetic model. This is far too expensive to be done in each step of our algorithms, since we are looking for an algorithm that has a total running time of $O(n^2)$. It is natural to try to approximate the norm, but since elements can have small norm but large representation, i.e. be unbalanced, there may be much cancellation during the computation of the norm. As far as we know there is no general method for handling cancellation.

We consider a weaker measure, which we call $N_+ : K \to \mathbb{R}$, and prove some results about this function. This function can not be computed efficiently, but in contrast to the norm it can be approximated within a constant factor.

6.7.1 The Positive Measure

Let $\alpha = a_0 + a_1\zeta + a_2\zeta^2 + a_3\zeta^3$ be an element in D. Since $\zeta = \frac{1}{2}\sqrt{2}(1+i)$ we can by substitution write

$$\alpha = \frac{1}{2}(A_{0,0}\sqrt{2} + A_{0,1}) + \frac{1}{2}(A_{1,0}\sqrt{2} + A_{1,1})i$$

where $A_{0,0}, A_{0,1}, A_{1,0}, A_{1,1} \in \mathbb{Z}$. This can be done in linear time since only a constant number of additions and subtractions are needed if we keep $\sqrt{2}$ as a formal symbol in the obvious way.

Definition 17. Let $\alpha \in D$ be expressed as described above. We define the *positive measure* $N_+ : K \to \mathbb{Q}$ by

$$N_+\alpha = \frac{1}{16}((|A_{0,0}|\sqrt{2} + |A_{0,1}|)^2 + (|A_{1,0}|\sqrt{2} + |A_{1,1}|)^2)^2 \ .$$

Remark 2. The positive measure is in fact unnecessarily complicated, since the outermost squaring and the $\frac{1}{4}$-factors do not carry any essential information. We use the more complicated definition, to allow easy comparison with the norm N.

It is not hard to see that N_+ is not a norm and in fact approximates the norm N arbitrarily badly. For example, $N_+(\epsilon^j)$ gets arbitrarily large for increasing j, whereas $N(\epsilon^j) = 1$ since ϵ^j is a unit. Our interest in the function N_+ is explained by the following results.

The first lemma says that if an element is Δ-balanced, then N_+ is a good approximation of the norm N.

Lemma 18. *Let α in D be Δ-balanced with $\Delta \geq 2$. Then*

$$N\alpha \leq N_+\alpha \leq \Delta^2 N\alpha \ .$$

Proof. The left inequality is obvious. For the right inequality there are two cases. Write $\alpha_3 = \frac{1}{2}(A'_{0,0}\sqrt{2} + A'_{0,1}) + \frac{1}{2}(A'_{1,0}\sqrt{2} + A'_{1,1})i$. Recall that $\sqrt{2} = \zeta - \zeta^3$ and note that it is kept fixed by σ_1 and σ_7, but $\sqrt{2}^{\sigma_3} = \sqrt{2}^{\sigma_5} = -\sqrt{2}$.

Suppose that $\text{sign}(A_{0,0}) = \text{sign}(A_{1,0})$ and $\text{sign}(A_{0,1}) = \text{sign}(A_{1,1})$. Then it follows from the action of σ_3 on $\sqrt{2}$ that $\text{sign}(A'_{0,0}) = \text{sign}(A'_{1,0})$ and $\text{sign}(A'_{0,1}) = \text{sign}(A'_{1,1})$, and that $\text{sign}(A_{0,0}) \neq \text{sign}(A'_{1,0})$. Thus, we have

$$N_+\alpha = \max\{|\alpha|^4, |\alpha_3|^4\} \leq \Delta^2 N\alpha \ ,$$

since α is Δ-balanced, and the claim follows.

Suppose that $\text{sign}(A_{0,0}) \neq \text{sign}(A_{1,0})$. Then the corresponding inequality holds also for α_3, and we have

$$\begin{aligned}(N_+\alpha)^{1/2} &\leq \frac{1}{2}\max\{(|A_{0,0}|\sqrt{2} + |A_{0,1}|)^2, (|A_{1,0}|\sqrt{2} + |A_{1,1}|)^2\} \\ &\leq 2|\alpha|^2, 2|\alpha_3|^2 \ .\end{aligned}$$

Thus, $N_+\alpha \leq 4N\alpha$, and the claim follows. □

For the above lemma to be useful there must be a way to balance an element α without computing its norm $N\alpha$. The next lemma says that $\epsilon^j\alpha$ is almost balanced when $N_+(\epsilon^j\alpha)$ is almost minimized over $j \in \mathbb{Z}$.

Lemma 19. *Let $0 < \Gamma < 1$ and let $\alpha \in D$. Choose $j_\alpha \in \mathbb{Z}$ such that $N_+(\epsilon^j\alpha) \geq \Gamma N_+(\epsilon^{j_\alpha}\alpha)$ for $j \neq j_\alpha$. Then $\epsilon^{j_\alpha}\alpha$ is $\left(\frac{\epsilon^4-1}{\epsilon^2\Gamma^{1/4}-1}\right)^{1/2}$-balanced.*

Lemma 19. First note that j_α is well defined since $N\alpha = N(\epsilon^j\alpha) \geq N_+(\epsilon^j\alpha)$ for all $j \in \mathbb{Z}$. Write $\alpha' = \epsilon^{j_\alpha}\alpha = \frac{1}{2}(A_{0,0}\sqrt{2} + A_{0,1}) + \frac{1}{2}(A_{1,0}\sqrt{2} + A_{1,1})i$.

Without loss we can assume that $A_{0,1}, A_{1,1} \geq 0$. Recall that $\sqrt{2} = \zeta - \zeta^3$, so $\sigma_3(\sqrt{2}) = -\sqrt{2}$. Thus, we have

$$\alpha'_3 = (\epsilon^{j_\alpha}\alpha)^{\sigma_3} = \frac{1}{2}(-A_{0,0}\sqrt{2} + A_{0,1}) + \frac{1}{2}(-A_{1,0}\sqrt{2} + A_{1,1})i \ .$$

There are two cases that we treat separately.

If $\text{sign}(A_{0,0}) = \text{sign}(A_{1,0})$, then either $|\alpha'|^4 = N_+\alpha'$ or $|\alpha'_3|^4 = N_+\alpha'$. Without loss we assume the former. We have $N_+(\alpha'/\epsilon^j) = N_+(\alpha')/\epsilon^j$ for all $j \in \mathbb{Z}$, since ϵ is a real number. This implies that

$|\alpha'/\epsilon^{j_\Gamma}|^4 < \Gamma N_+\alpha'$ when $j_\Gamma > \frac{1}{4}\log_\epsilon(1/\Gamma)$. This implies that we must have $|\alpha'_3|^4\epsilon^{4j_\Gamma} = N_+(\epsilon\alpha')$ and by construction of α' we also have $N_+(\epsilon\alpha') \geq \Gamma N_+\alpha'$. We conclude that

$$|\alpha'|/\epsilon^{4j_\Gamma} < \Gamma N_+\alpha' \leq \epsilon^{4j_\Gamma}|\alpha'_3|^4$$

which implies that that $\frac{|\alpha'|}{|\alpha'_3|} \leq \epsilon^{2j_\Gamma} < \frac{1}{\Gamma^{1/2}}$ as claimed.

If $\text{sign}(A_{0,0}) \neq \text{sign}(A_{1,0})$, then we assume without loss that $|A_{0,0}|\sqrt{2}+|A_{0,1}| \leq |A_{1,0}|\sqrt{2}+|A_{1,1}|$ and consider

$$\begin{aligned} 4(N_+(\epsilon\alpha))^{1/4} &= \epsilon^2(|A_{0,0}|\sqrt{2}+|A_{0,1}|)^2 + \frac{1}{\epsilon^2}(|A_{1,0}|\sqrt{2}+|A_{1,1}|)^2 \\ &\geq \Gamma^{1/4}\left((|A_{0,0}|\sqrt{2}+|A_{0,1}|)^2 + (|A_{1,0}|\sqrt{2}+|A_{1,1}|)^2\right) \ , \end{aligned}$$

which holds from the construction of α'. This inequality implies that

$$(|A_{1,0}|\sqrt{2}+|A_{1,1}|)^2 \leq \frac{\epsilon^2-\Gamma^{1/4}}{\Gamma^{1/4}-\epsilon^{-2}}(|A_{0,0}|\sqrt{2}+|A_{0,1}|)^2 \ .$$

Suppose that $A_{1,0} \leq 0$. Then

$$\begin{aligned} 4|\alpha'|^2 &\leq 4(N_+\alpha')^{1/2} \leq \left(1+\frac{\epsilon^2-\Gamma^{1/4}}{\Gamma^{1/4}-\epsilon^{-2}}\right)(|A_{0,0}|\sqrt{2}+|A_{0,1}|)^2 \\ &\leq 4\left(1+\frac{\epsilon^2-\Gamma^{1/4}}{\Gamma^{1/4}-\epsilon^{-2}}\right)|\alpha'_3|^2 = 4\frac{\epsilon^4-1}{\epsilon^2\Gamma^{1/4}-1}|\alpha'_3|^2 \ . \end{aligned}$$

The case for $A_{0,0} \leq 0$ is similar. □

Loosely speaking these results guarantee that if we can compute N_+ efficiently and are given an element that is already fairly balanced, we can balance the element and efficiently compute an approximation of $N\alpha$ within a constant factor $\left(\frac{\epsilon^4-1}{\epsilon^2\Gamma^{1/4}-1}\right)^{1/2}$ efficiently.

Unfortunately, we can not compute N_+ quickly. But in contrast to the norm N, it is not hard to approximate N_+ within a constant factor.

6.7.2 Approximating the Positive Measure

In this section we explain how to approximate N_+ within a constant factor in time $O(\log n)$. We have the following lemma.

Lemma 20. *Let $0 < \Gamma < 1$ be constant. Given $\alpha \in D$, an approximation $N_+^\Gamma\alpha$ of $N_+\alpha$ can be computed in time $O(\log n)$, such that*

$$\Gamma N_+\alpha \leq N_+^\Gamma\alpha \leq N_+\alpha \ .$$

The idea is simple. Since computing $N_+\alpha$ only involves multiplications and additions of *positive* numbers, we can compute with constant precision of only w bits, say $w = 16$, and get a result within a constant factor of the true value of $N_+\alpha$. By using higher precision we can make the constant factor arbitrarily small. We give the details of this in Appendix A.

7 The Algorithms

We are now ready to give the algorithms. The general idea is to extend the l-ary gcd algorithm for integers to the ring D. We first describe two subroutines used by both algorithms. Then we describe the main algorithms, but without all details. Our objective is to emphasize the similarity with the l-ary algorithm of Sorenson [7], and to improve readability. Finally, we give full details on how each step in the algorithm is computed.

In the text above we have stated several results with parameters. Computing with w-bit precision when we approximate N_+ gives an approximation N_+^Γ, where $\Gamma(w) = (1 + 2^{1-w})^{-30}$. This follows from Appendix A. Given Γ we define

$$\begin{aligned} \Delta(\Gamma) &= \left(\frac{\epsilon^4 - 1}{\epsilon^2 \Gamma^{1/4} - 1} \right)^{1/2} \\ t(l) &= 2 + \frac{2}{\log_2 l} \ . \end{aligned}$$

For the algorithm to work, we require that Γ and l are chosen such that

$$64\epsilon^2 \Delta(\Gamma)^2 (1 + \Delta(\Gamma))^2 l^{t(l)} < l^4 \ .$$

Using 16-bit precision in the computation of N_+^Γ, we need $l \geq 365$. The necessary value of l does not decrease notably with greater Γ when Γ is already close to 1. Here we see that it is useful to have as small $t(l)$ as possible (cf. Remark 1).

7.1 Common Subroutines

In this section we describe subroutines invoked by the main algorithms. We state the main algorithms in terms of subroutine calls to improve readability and simplify analysis.

Algorithm 1 (Balance Element).
BALANCE(α)
INPUT: $\alpha \in D$.
OUTPUT: (α', j_α), where $\alpha' = \epsilon^{j_\alpha} \alpha$ is Δ-balanced.

1. Set $\alpha_0 = \alpha$, and $j = 0$.
2. If $N_+^\Gamma \alpha_j \leq N_+^\Gamma(\epsilon \alpha_j)$ and $N_+^\Gamma \alpha_j \leq N_+^\Gamma(\epsilon^{-1} \alpha_j)$, return (α_j, j).
3. If the first inequality is not satisfied, set $\alpha_j = \epsilon^{-1} \alpha_{j-1}$ and $j = j - 1$. If the second inequality is not satisfied, set $\alpha_j = \epsilon \alpha_{j-1}$ and $j = j + 1$. Then go to Step 2.

Lemma 21. *The output of the* BALANCE*-algorithm is Δ-balanced, and the algorithm runs in time $O(n|\log \frac{|\alpha|}{|\alpha_3|}|)$.*

Proof. From Lemma 20 we know that the algorithm finds an j_α such that the inequality $N_+(\epsilon^j\alpha) \geq \Gamma N_+(\epsilon^{j_\alpha}\alpha)$ for $j \neq j_\alpha$. Lemma 19 then implies that $\epsilon^{j_\alpha}\alpha$ is Δ-balanced. □

Consider the set of non-unit elements that divide some $\delta \in T_{l,t}$. This set is clearly infinite, since each element in D has an infinite number of associates. This makes it natural to consider the following set instead

$$\begin{aligned} F_{l,t} &= \{\omega \in D \ : \ \omega \mid \delta,\, \delta \in T_{l,t},\, \omega \text{ is non-unit, primary, prime and balanced}\} \\ &\quad \cup\{l, 1+\zeta, 1-\zeta\} \ . \end{aligned}$$

Note that we include l, which has $Nl = l^4$, in $F_{l,t}$. We also include $1+\zeta$ and $1-\zeta$ which are prime but with even norm. This is notationally convenient. The set $F_{l,t}$ is bounded and we denote its elements by $F_{l,t} = \{\omega_1, \ldots, \omega_{s_F}\}$. We write $F_{l,t} \nmid \alpha$ to denote the fact that $\omega \nmid \alpha$ for all $\omega \in F_{l,t}$.

Algorithm 2 (Extracting Small Factors).
SMALL(α)
INPUT: $\alpha \in D$.
OUTPUT: $(\alpha', (j_1, \ldots, j_{s_F}))$, where $\alpha = \alpha' \prod_{j=1}^{s_F} \omega_j$ and $F_{l,t} \nmid \alpha'$.

The algorithm is the trivial one. Find α', and j_k by trial division.

Lemma 22. *Let $\alpha \in D$ and suppose $(\alpha', (j_1, \ldots, j_{s_F})) =$ SMALL(α). Then the running time of the* SMALL*-algorithm on input α is $O((\sum_{k=1}^{s_F} j_k)n)$.*

Proof. Trial division by an integer is done by trial division with all coefficients a_i of an element $\alpha = a_0 + a_1\zeta + a_2\zeta^2 + a_3\zeta^3$. This can obviously be done in time $O(n)$.

Note that $\gamma \mid \alpha$ if and only if $N\gamma \mid \frac{N\gamma}{\gamma}\alpha$. Thus, trial division by an element in D is reduced to trial division by an integer in time $O(n)$.

It now suffices to note that $|F_{l,t}|$ is constant and each trial division can be carried out in time $O(n)$. □

7.2 Greatest Common Divisor

The main algorithm first calls the SMALL-algorithm to find all common factors from the set $F_{l,t}$. Then it invokes a subroutine IGCD, which assumes inputs without such factors. In each call to the subroutine IGCD it strips its first input from all factors in $F_{l,t}$. Then it finds a linear combination with small factors that is divisible by l and calls itself recursively.

Algorithm 3 (Greatest Common Divisor).
GCD(α, β)
INPUT: $\alpha, \beta \in D$
OUTPUT: The greatest common divisor of α and β.

1. Compute $(\alpha_0, (j_1, \ldots, j_{s_F})) = \text{SMALL}(\alpha)$ and $(\beta_0, (j'_1, \ldots, j'_{s_F})) = \text{SMALL}(\beta)$.
2. Compute $(\alpha_1, \cdot) = \text{BALANCE}(\alpha_0)$ and $(\beta_1, \cdot) = \text{BALANCE}(\beta_0)$.
3. Compute $(\lambda, \cdot) = \text{SMALL}(\text{IGCD}(\alpha_1, \beta_1))$.
4. Return $\lambda \prod_{k=1}^{s_F} \omega_k^{\min\{j_k, j'_k\}}$.

Algorithm 4.
IGCD(α, β)
INPUT: $\alpha, \beta \in D$ where $F_{l,t} \nmid \beta$.
OUTPUT: $\gcd(\alpha, \beta) \prod_{k=1}^{s_F} \omega_k^{j''_k}$ for some j''_k.

1. If $\alpha = 0$ return β.
2. Compute $(\alpha_0, \cdot) = \text{BALANCE}(\alpha)$.
3. If $N_+^\Gamma \alpha_0 > N_+^\Gamma \beta$ then set $(\alpha_1, \beta_1) = (\beta, \alpha_0)$ and otherwise $(\alpha_1, \beta_1) = (\alpha_0, \beta)$.
4. Compute $\gamma, \delta \in T_{l,t}$ such that $l \mid (\gamma\alpha_1 + \delta\beta_1)$.
5. Return IGCD$((\gamma\alpha_1 + \delta\beta_1)/l, \beta_1)$.

7.3 Octic Residue Symbol

The algorithm for the octic residue symbol is similar to the GCD-algorithm, but in each step we keep track of the octic symbol of the factors that are factored out from α. This becomes quite complex, but it is essentially just repeated application of the reciprocity laws, Theorem 4 and Theorem 5. Readers interested in the precise details of the calculations are referred to Section 7.4 and the proof of Proposition 24 below.

Algorithm 5 (Octic Residue Symbol).
OCTIC(α, β)
INPUT: $\alpha, \beta \in D$, where β is a primary non-unit with $N\beta = 1 \pmod 8$.
OUTPUT: The octic residue symbol $\left(\frac{\alpha}{\beta}\right)$ of α modulo β.

1. Compute $(\alpha_0, (j_1, \ldots, j_{s_F})) = \text{SMALL}(\alpha)$ and $(\beta_0, (j'_1, \ldots, j'_{s_F})) = \text{SMALL}(\beta)$.

2. If $j_k > 0$ and $j'_k > 0$ for any k, return 0.

3. Compute $(\alpha_1, j_\alpha) = \text{BALANCE}(\alpha_0)$, and set

$$s_0 = \left(\frac{\epsilon}{\beta}\right)^{-j_\alpha} \prod_{k=1}^{s_F} \left(\frac{\omega_k}{\beta}\right)^{j_k} .$$

4. If α_1 is a unit, return $s_0 \left(\frac{\alpha_1}{\beta}\right)$.

5. Compute $j_\epsilon, j_\zeta \in \{0, 1, 2, 3\}$ such that $\alpha_2 = \epsilon^{j_\epsilon} \zeta^{j_\zeta} \alpha_1$ is primary and set

$$s_1 = \left(\left(\frac{\epsilon}{\beta}\right)^{j_\epsilon} \left(\frac{\zeta}{\beta}\right)^{j_\zeta}\right)^{-1} .$$

6. Compute s_2 such that $\left(\frac{\alpha_2}{\beta}\right) = s_2 \left(\frac{\beta}{\alpha_2}\right)$.

7. Compute $(\beta_1, j_\beta) = \text{BALANCE}(\beta_0)$, and set

$$s_3 = \left(\frac{\epsilon}{\alpha_2}\right)^{-j_\beta} \prod_{k=1}^{s_F} \left(\frac{\omega_k}{\alpha_2}\right)^{j_k} .$$

8. Return $s_0 s_1 s_2 s_3 \text{IOCTIC}(\beta_1, \alpha_2)$.

Algorithm 6.
IOCTIC(α, β)
INPUT: $\alpha, \beta \in D$, where β is primary and $F_{l,t} \nmid \beta$.
OUTPUT: The octic residue symbol $\left(\frac{\alpha}{\beta}\right)$ of α modulo β.

1. If $\alpha = 0$ return 0.

2. Compute $(\alpha_0, (j_1, \ldots, j_{s_F})) = \text{SMALL}(\alpha)$ and set

$$s_0 = \prod_{k=1}^{s_F} \left(\frac{\omega_k}{\beta}\right)^{j_k} .$$

3. Compute $(\alpha_1, j_\alpha) = \text{BALANCE}(\alpha_0)$, and set

$$s_1 = \left(\frac{\epsilon}{\beta}\right)^{-j_\alpha} .$$

4. If α_1 is a unit, return $s_0 s_1 \left(\frac{\alpha_1}{\beta}\right)$.

5. Compute $j_\epsilon, j_\zeta \in \{0,1,2,3\}$ such that $\alpha_2 = \epsilon^{j_\epsilon}\zeta^{j_\zeta}\alpha_1$ is primary and set

$$s_2 = \left(\left(\frac{\epsilon}{\beta}\right)^{j_\epsilon}\left(\frac{\zeta}{\beta}\right)^{j_\zeta}\right)^{-1} .$$

6. If $N_+^\Gamma \alpha_1 < N_+^\Gamma \beta$ then set $(\alpha_3, \beta_3) = (\beta, \alpha_2)$ and otherwise $(\alpha_3, \beta_3) = (\alpha_2, \beta)$. Then compute s_3 such that

$$\left(\frac{\alpha_2}{\beta}\right) = s_3\left(\frac{\alpha_3}{\beta_3}\right) .$$

7. Compute $\gamma, \delta \in T_{l,t}$ such that $l \mid (\gamma\alpha_3 + \delta\beta_3)$ and set

$$s_4 = \left(\frac{\gamma}{\beta_3}\right)^{-1}$$

8. Return $s_0 s_1 s_2 s_3 s_4 \mathrm{IOCTIC}(\gamma\alpha_3 + \delta\beta_3, \beta_3)$.

Remark 3. In the octic residue algorithm it seems necessary to include a call to SMALL in every call made by IOCTIC. The problem is that if intermediate α and β are not relatively prime, then we can not safely apply the reciprocity law.

7.4 Details of the Algorithms

The description of the GCD-algorithm is fairly easy to understand, but it may not be clear to all readers how each step of the OCTIC-algorithm is computed.

Consider the evaluation of expressions on the form $\left(\frac{\omega_k}{\beta}\right)$. There are several cases. If ω_k equals $1+\zeta$ or $1-\zeta$. Then we apply Proposition 5 to compute $\left(\frac{\omega_k}{\beta}\right)$. If $\omega_k = l$ we write $l = (1+\zeta)^{l_1}(1-\zeta)^{l_2}l'$ where $1+\zeta, 1-\zeta \nmid l'$. Then we apply Proposition 3 and conclude $\left(\frac{l}{\beta}\right) = \left(\frac{1+\zeta}{\beta}\right)^{l_1}\left(\frac{1+\zeta}{\beta}\right)^{l_1}\left(\frac{l'}{\beta}\right)$. Theorem 4 then gives an s such that $\left(\frac{l'}{\beta}\right) = s\left(\frac{\beta}{l'}\right)$. The last residue equals $\left(\frac{\beta \bmod l'}{l'}\right)$, and there are at most a constant number of such residues, so this can be computed in constant time.

Otherwise, $N\omega_k$ is primary. Thus, $\left(\frac{\omega_k}{\beta}\right) = s\left(\frac{\beta}{\omega_k}\right) = \left(\frac{\beta \bmod N\omega_k}{\omega_k}\right)$ for some s computed in constant time using Theorem 4. There are a constant number of residues of the last type since $N\omega_k$ is an integer. Thus, this residue can be computed in constant time.

Consider next expressions on the forms $\left(\frac{\epsilon}{\beta}\right)$ and $\left(\frac{\zeta}{\beta}\right)$. Such expressions can be evaluated using Theorem 5. Again we need only compute modulo 8, so this can be done in constant time.

In Step 4 of IOCTIC we must check if an element α_1 is a unit. Since α_1 is Δ-balanced we may invoke Lemma 12 and conclude that we need only check if α_1 is one of the constant number of possible Δ-balanced units. If it is not, we know that it is a non-unit.

Next we consider how to perform Step 6. From Lemma 20 we know that we can compute $N_+^\Gamma \alpha_1$ and $N_+^\Gamma \beta$ in time $O(\log n)$. The value of s_3 can be computed in constant time using Theorem 4, since we need only compute modulo 8.

To find suitable γ and δ in Step 7 we only need to compute modulo l. Reducing α and β modulo l can be done in time $O(n)$. Then finding γ and δ is done in constant time.

The residue symbol $\left(\frac{\gamma}{\beta_3}\right)$ is computed by factoring $\gamma = \prod_{k=1}^{s_\gamma} \omega_k^{j_k}$, applying Proposition 3, i.e. $\left(\frac{\gamma}{\beta_3}\right) = \prod_{k=1}^{s_\gamma} \left(\frac{\omega_k}{\beta_3}\right)^{j_k}$, and computing each of these symbols as outlined above.

This completes the description of the algorithms.

8 Analysis

In this section we analyze the algorithms. First we prove correctness of each algorithm provided they halt at all. Then we bound the running time of the algorithms.

Proposition 23. *If the* GCD*-algorithm halts, it outputs the greatest common divisor of its inputs α and β.*

Proof. Suppose that $\gcd(\alpha, \beta) = \lambda^*$, and consider the GCD algorithm. By Lemma 21 and the definition of the BALANCE-algorithm we have $\alpha = \alpha_1 \gamma \prod_{k=1}^{s_F} \omega_k^{j_k}$ and $\beta = \beta_1 \delta \prod_{k=1}^{s_F} \omega_k^{j'_k}$ for some units γ, δ, where $F_{l,t} \nmid \alpha_1$ and $F_{l,t} \nmid \beta_1$.

Thus, if $\lambda = \gcd(\alpha_1, \beta_1)$, then $\lambda^* = \lambda\alpha \prod_{k=1}^{s_F} \omega_k^{\min\{j_k, j'_k\}}$. By Lemma 21 this holds if $\mathrm{IGCD}(\alpha_1, \beta_1)$ is on the form $\gcd(\alpha_1, \beta_1) \prod_{k=1}^{s_F} \omega_k^{j''_k}$ for some j''_k. We conclude that GCD is correct if IGCD is correct.

If $\alpha = 0$ the gcd is β and if α is a unit the gcd is a unit, so in this case IGCD is correct. Since $\alpha_0 = \alpha\psi$ for some unit ψ we have $\gcd(\alpha_0, \beta) = \gcd(\alpha, \beta)$. Changing the order of α_0 and β does not change the gcd. Finally, by Lemma 16, we know that $\gcd((\gamma\alpha_1 + \delta\beta_1)/l, \beta_1) = \gcd(\alpha_1, \beta_1) \prod_{k=1}^{s_F} \omega_k^{j''_k}$ for some j''_k, so the IOCTIC-algorithm is correct if it halts.

This implies that the OCTIC-algorithm is correct if it halts. □

Proposition 24. *If the* OCTIC*-algorithm halts, it outputs the octic residue symbol of α modulo β.*

Proof. If α and β has a common factor in $F_{l,t}$ this is discovered in Step 2 of the OCTIC-algorithm, and the output is 0, which is correct.

In general, when we turn a residue symbol on its head, we always ensure that both operands are primary. By Lemma 9 this implies that their norms equal 1 modulo 8, i.e. both residue symbols are well defined.

Suppose that $\mathrm{IOCTIC}(\beta_1, \alpha_2) = \left(\frac{\beta_1}{\alpha_2}\right)$. Then we have

$$\begin{aligned} s_0 s_1 s_2 s_3 \mathrm{IOCTIC}(\beta_1, \alpha_2) &= s_0 s_1 s_2 \left(\frac{\epsilon}{\alpha_2}\right)^{-j_\beta} \prod_{k=1}^{s_F} \left(\frac{\omega_k}{\alpha_2}\right)^{j_k} \left(\frac{\beta_1}{\alpha_2}\right) \\ &= s_0 s_1 s_2 \left(\frac{\beta}{\alpha_2}\right) = s_0 s_1 \left(\frac{\alpha_2}{\beta}\right) \\ &= s_0 \left(\left(\frac{\epsilon}{\beta}\right)^{j_\epsilon} \left(\frac{\zeta}{\beta}\right)^{j_\zeta}\right)^{-1} \left(\frac{\alpha_2}{\beta}\right) \\ &= s_0 \left(\frac{\alpha_1}{\beta}\right) = \left(\frac{\epsilon}{\beta}\right)^{-j_\alpha} \prod_{k=1}^{s_F} \left(\frac{\omega_k}{\beta}\right)^{j_k} \left(\frac{\alpha_1}{\beta}\right) \\ &= \left(\frac{\alpha}{\beta}\right) \ . \end{aligned}$$

Here we use Theorem 4 and Proposition 3 repeatedly. We conclude that OCTIC is correct if IOCTIC is correct.

We prove this similarly to the above. If $\alpha = 0$ the output is 0 which is correct. Otherwise we note that

$$s_0 s_1 \left(\frac{\alpha_1}{\beta}\right) = s_0 \left(\frac{\epsilon}{\beta}\right)^{-j_\alpha} \left(\frac{\alpha_1}{\beta}\right) = \prod_{k=1}^{s_F} \left(\frac{\omega_k}{\beta}\right)^{j_k} \left(\frac{\alpha_0}{\beta}\right) = \left(\frac{\alpha}{\beta}\right) \ .$$

Thus, if α_1 is a unit, the result is correct, and otherwise we need only show that $s_2 s_3 s_4 \mathrm{IOCTIC}(\gamma\alpha_3 + \delta\beta_3, \beta_3) = \left(\frac{\alpha_1}{\beta}\right)$.

This can be seen as follows.

$$\begin{aligned} s_2 s_3 s_4 \left(\frac{\gamma\alpha_3 + \delta\beta_3}{\beta_3}\right) &= s_2 s_3 \left(\frac{\gamma}{\beta_3}\right)^{-1} \left(\frac{\gamma\alpha_3}{\beta_3}\right) = s_2 s_3 \left(\frac{\alpha_3}{\beta_3}\right) \\ &= s_2 \left(\frac{\alpha_2}{\beta}\right) = \left(\left(\frac{\epsilon}{\beta}\right)^{j_\epsilon} \left(\frac{\zeta}{\beta}\right)^{j_\zeta}\right)^{-1} \left(\frac{\alpha_2}{\beta}\right) = \left(\frac{\alpha_1}{\beta}\right) \ . \end{aligned}$$

Thus, the algorithm is correct if $\mathrm{IOCTIC}(\gamma\alpha_3 + \delta\beta_3, \beta_3) = \left(\frac{\alpha_1}{\beta}\right) = \left(\frac{\gamma\alpha_3 + \delta\beta_3}{\beta_3}\right)$, and we are done. □

At this point we have established that the algorithms are correct if they halt at all. All that remains is to analyze their complexity. We only analyze the OCTIC-algorithm. It is easy to see that the running time of this algorithm strictly bounds the running time of the GCD-algorithm.

Lemma 25. *Consider α and $\alpha_0 \neq 0$ in an invocation of* IOCTIC. *The running time of the invocation excluding the running time of the recursive call is $O(n \log \frac{N\alpha}{N\alpha_0})$.*

Proof. First note that in Section 7.4 above we show that if we ignore the running time of the two subroutine calls, an invocation runs in time $O(n)$. Next we consider the two subroutine calls.

Recall that $\alpha = \alpha_0 \prod_{k=1}^{s_F} \omega_k^{j_k}$. Taking the norm on both sides and dividing by $N\alpha_0$ gives

$$\frac{N\alpha}{N\alpha_0} = \prod_{k=1}^{s_F} N(\omega_k)^{j_k} \ .$$

We know that $N\omega \geq 2$ for all $\omega \in F_{l,t}$, since ω is prime in D which implies that $N\omega$ is prime in $\mathbb{Z}$. Thus,

$$\log_2 \frac{N\alpha}{N\alpha_0} = \sum_{k=1}^{s_F} j_k \log_2 N(\omega_k) \geq \sum_{k=1}^{s_F} j_k \ .$$

Lemma 21 now implies that the claim is true if we ignore the cost of the invocation of the BALANCE-algorithm. Thus, it now suffices to prove that this invocation runs in time $O(n \log \frac{N\alpha}{N\alpha_0})$ as well.

We have $N\alpha = |\alpha|^2 |\alpha^{\sigma_3}|^2$. Without loss we assume $|\alpha| \geq |\alpha^{\sigma_3}|$ and $|\alpha_0| \geq |\alpha_0^{\sigma_3}|$, since this is one of the two symmetric difficult cases. We have $|\alpha_0| \leq |\alpha|$ since dividing by primes can only decrease the complex absolute value. We now have

$$\frac{|\alpha_0|^4}{|\alpha_0^{\sigma_3}|^4} \leq \frac{|\alpha|^4}{|\alpha_0^{\sigma_3}|^4} \leq \frac{\Delta^2 N\alpha}{|\alpha_0^{\sigma_3}|^4} = \frac{\Delta^2 N\alpha \cdot |\alpha_0|^4}{|\alpha_0|^4 \cdot |\alpha_0^{\sigma_3}|^4} \leq \frac{(\Delta^2 N\alpha)^2}{(N\alpha_0)^2} \ .$$

The claim now follows from Lemma 21, since the BALANCE-algorithm runs in time $O(n \log \frac{|\alpha_0|}{|\alpha_0^{\sigma_3}|})$ and Δ is constant. □

Write $\alpha^{(j)}$ to denote the value of α in the jth recursive call to IOCTIC, and similarly for the other variables.

Lemma 26. *For $j > 1$ we have*

$$\begin{aligned} N\alpha^{(j)} &\leq (64\epsilon^2\Delta^2(1+\Delta)^2 l^t) N\alpha_0^{(j-2)} \ , \quad \textit{and} \\ N\alpha_0^{(j)} &\leq \left(\frac{64\epsilon^2\Delta^2(1+\Delta)^2 l^t}{l^4}\right) N\alpha_0^{(j-2)} \ . \end{aligned}$$

Proof. From Lemma 21 we know that $\alpha_1^{(j)}$ is Δ-balanced. By construction, $N\alpha_1^{(j)} = N\alpha_0^{(j)}$. It follows from Lemma 15 and the fact that $l \mid \alpha^{(j)}$ (except for the the very first call) that

$$\begin{aligned} N\alpha^{(j)} &\leq (64\epsilon^2(1+\Delta)^2 l^t)\max\{N\alpha_0^{(j-1)}, N\beta^{(j-1)}\}, \quad \text{and} \\ N\alpha_0^{(j)} &\leq \frac{64\epsilon^2(1+\Delta)^2 l^t}{l^4}\max\{N\alpha_0^{(j-1)}, N\beta^{(j-1)}\} \ , \end{aligned} \tag{1}$$

By construction and Lemma 18 we also have

$$N\beta^{(j-1)} \leq \Delta^2 N\alpha_1^{(j-2)} = \Delta^2 N\alpha_0^{(j-2)} \ .$$

Thus, it remains to show that $N\alpha_0^{(j-1)} \leq \Delta^2 N\alpha_0^{(j-2)}$ when $N\alpha_0^{(j-1)} \geq N\beta^{(j-1)}$.

By translating Equation 1 above we have

$$N\beta^{(j-1)} \leq N\alpha_0^{(j-1)} \leq \frac{64\epsilon^2(1+\Delta)^2 l^t}{l^4}\max\{N\alpha_0^{(j-2)}, N\beta^{(j-2)}\} \ ,$$

but when $N\alpha_0^{(j-1)} \geq N\beta^{(j-1)}$ we must have $N\beta^{(j-2)} \leq \Delta^2 N\beta^{(j-1)}$ and l was chosen such that $\frac{64\epsilon^2\Delta^2(1+\Delta)^2 l^t}{l^4} < 1$. Thus, we must have $\max\{N\alpha_0^{(j-2)}, N\beta^{(j-2)}\} = N\alpha_0^{(j-2)}$ which concludes the proof. □

Proposition 27. *The* GCD*-algorithm and the* OCTIC*-algorithm run in time* $O(n^2)$.

Proof. As explained above it suffices to show that the running time of the OCTIC-algorithm is $O(n^2)$.

It is easy to see that the computations done in OCTIC is done in time $O(n^2)$ if we ignore the call to IOCTIC. The interesting part is what happens in each call to IOCTIC.

Set $C = 64\epsilon^2\Delta^2(1+\Delta)^2 l^t$. The second inequality of Lemma 26 implies that every other recursive call $\max\{N\alpha, N\beta\}$ has been reduced at least by a factor $C/l^4 < 1$. Thus, the IOCTIC-algorithm makes at most $m = O(n)$ recursive calls. We ignore the cost of the very last call if $\alpha_0^{(m)} = 0$, since no computations are done in this case. This allows us to assume that $N\alpha_0^{(j)} \geq 1$ for all j.

We can now bound the running time $T(n)$ of the algorithm by

$$T(n) = \sum_{j=1}^{m} O\left(n \log \frac{N\alpha^{(j)}}{N\alpha_0^{(j)}}\right) = O\left(n \log\left(\prod_{j=1}^{m} \frac{N\alpha^{(j)}}{N\alpha_0^{(j)}}\right)\right) \ .$$

Note that we needed to show that the algorithms halts for the above expression to have meaning.

Then we apply the first inequality of Lemma 26 and get

$$\begin{aligned}\prod_{j=1}^{m}\frac{N\alpha^{(j)}}{N\alpha_0^{(j)}} &\leq N\alpha_0^{(1)}N\alpha_0^{(2)}\prod_{j=3}^{m}\frac{CN\alpha_0^{(j-2)}}{N\alpha_0^{(j)}} \leq C^{m-2}\frac{N\alpha_0^{(1)}N\alpha_0^{(2)}}{N\alpha_0^{(m-1)}N\alpha_0^{(m)}} \\ &\leq C^{m-2}\max\{N\alpha^{(1)},N\beta^{(1)}\}^2 \ .\end{aligned}$$

This implies that

$$T(n)=O\left(n\log\left(C^{m+2}\max\{N\alpha^{(1)},N\beta^{(1)}\}\right)^2\right)=O(n(m+2+n))=O(n^2) \ ,$$

which concludes the proof. □

9 Further Applications

No essential changes are needed to give gcd algorithms similar to our gcd algorithm for the ring of integers $O_d=\mathbb{Z}+\mathbb{Z}w$ in any quadratic number field $\mathbb{Q}(\sqrt{d})$. In fact, our algorithm can be used directly when $d=-1,2,-2$, since $\mathbb{Q}(i)$, $\mathbb{Q}(\sqrt{2})$, and $\mathbb{Q}(\sqrt{-2})$ are subfields of $\mathbb{Q}(\zeta)$. For other d our claim follows since there is at most one fundamental unit ϵ_d in O_d, and we may define $N_+(a_0+a_1w)=(|a_0|+|a_1||w|)^2$, which is easily approximated. This gives a result similar to that of Kaltofen and Rolletschek [4] but using the l-ary approach.

Furthermore, we see no reason why our approach would not work in the ring of integers of any biquadratic extension of $\mathbb{Q}$ which has only one fundamental unit. In particular it is possible to compute the gcd in $\mathbb{Z}[\zeta_5]$, where ζ_5 is a primitive fifth root of unity. It is also be possible to compute the quintic residue symbol.

10 Future Work

An interesting open problem is to what extent the l-ary algorithm can be generalized further. There are two main problems. One must be able to balance elements quickly when there are several fundamental units, and there must be a function with similar properties as N_+, which can be approximated quickly.

Another interesting problem is to determine if l, which is constant in our algorithm, can be made an increasing function of the input size as in Sorenson [7]. It seems difficult to do this when computing the residue symbol, since we can not allow the two elements to have any common factors if we want to use the reciprocity law to a symbol on its head, but for the gcd algorithm it is probably possible.

11 Acknowledgments

I am most grateful to my supervisor Johan Håstad for many fruitful suggestions and remarks. I am also grateful to Torsten Ekedahl who essentially played the role of an extra advisor during this work.

References

[1] S. Agarwal, G. Skjovbjerg Frandsen, *Binary GCD Like Algorithms for Some Complex Quadratic Rings*, ANTS 2004, LNCS 3076, pp. 57-71, 2004.

[2] I. Damgård, G. Skjovbjerg Frandsen, *Efficient Algorithms for gcd and Cubic Residuosity in the Ring of Eisenstein Integers*, BRICS Technical Report, ISSN 0909-0878, BRICS RS 03-8, 2003.

[3] K. Ireland, M. Rosen, *A Classical Introduction to Modern Number Theory*, 2nd edition 5:th printing, Springer-Verlag 1998, ISBN 0-387-97329-5.

[4] E. Kaltofen, H. Rolletschek, *Computing greatest common divisors and factorizations in quadratic number fields*, Mathematics of Computation, 53(188):697-720, 1989.

[5] F. Lemmermeyer, *Reciprocity Laws*, ISBN 3-540-66957-4, Springer-Verlag, 2000.

[6] J. Shallit, J. Sorenson, *A binary algorithm for the Jacobi symbol*, ACM SIGSAM Bulletin, 27 (1), pp. 4-11, 1993.

[7] J. Sorenson, *Two Fast GCD Algorithms*, Journal of Algorithms, 16(1):110-144, 1994.

[8] J. Stein, *Computational problems associated with Racah algebra*, Journal of Computational Physics No. 1, pp. 397-405, 1969.

[9] L. Washington, *Introduction to Cyclotomic Fields*, ISBN 0-387-90622-3, Springer-Verlag New York, 1982.

[10] A. Weilert, *Asymptotically fast GCD computation in* $\mathbb{Z}[i]$, In Algorithmic number theory (Leiden, 2000), LNCS 1838, pp. 595-613, 2000.

[11] A. Weilert, $(1+i)$*-ary GCD computation in* $\mathbb{Z}[i]$ *as an analogue to the binary GCD algorithm*, Journal of Symbolic Computation, 30(5):605-617, 2000.

[12] D. Wikström, *On the Security of Mix-Nets and Related Problems*, Licentiate thesis, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, TRITA NA 04-06, ISSN: 0348-2952, ISRN KTH/NA/R--04/06--SE, ISBN 91-7283-717-9, May, 2004.

A Details of the Approximation of the Positive Measure

Given a positive number $H = \sum_{j=-\infty}^{h} h_j 2^j$ in binary basis, i.e. $h_j \in \{0,1\}$, we denote by $\widetilde{H}$ the number $\sum_{j=h-w}^{h} h_j 2^j$. Thus, $\widetilde{H}$ is simply H, but truncated to w-bit precision. Similarly $\widetilde{\sqrt{2}}$ is $\sqrt{2}$ truncated to w-bit precision.

We define addition and multiplication on such numbers by $\widetilde{H} \oplus \widetilde{H} = \widetilde{\widetilde{H} + \widetilde{H}}$ and $\widetilde{H} \odot \widetilde{H} = \widetilde{\widetilde{H}\widetilde{H}}$, i.e. we simply compute with precision w. We also define exponentiation with precision w in the natural way by $\widetilde{H}^{\odot(s)} = \prod_{j=1}^{s} \widetilde{H}$.

We have the following lemma.

Lemma 28. *Let $H, H' \geq 2^w$ be positive integers, and define $\Theta = \Theta(w) = 1 + 2^{1-w}$. Then*

$$\begin{aligned} \frac{1}{\Theta} H \leq \quad & \widetilde{H} \quad \leq H \ , \\ \frac{1}{\Theta^2}(H + H') \leq \quad & \widetilde{H} \oplus \widetilde{H'} \quad \leq H + H' \quad , \quad \textit{and} \\ \frac{1}{\Theta^3} H H' \leq \quad & \widetilde{H} \odot \widetilde{H'} \quad \leq H H' \ . \end{aligned}$$

Proof. The right inequalities are obvious. For the first left inequality we have $H = \sum_{j=h-w+1}^{h} h_j 2^j + \sum_{j=-\infty}^{h-w} h_j 2^j \leq \widetilde{H} + 2^{h-w+1} \leq (1 + 2^{1-w})\widetilde{H} = \Theta\widetilde{H}$. The other inequalities follows by repeated application of the first. We have $H + H' \leq \Theta(\widetilde{H} + \widetilde{H'}) \leq \Theta^2(\widetilde{H} \oplus \widetilde{H'})$ and $HH' \leq \Theta^2 \widetilde{H}\widetilde{H'} \leq \Theta^3(\widetilde{H} \odot \widetilde{H'})$. □

We are finally ready to define the measure $N_+^{\Theta(w)}$ of the size of elements that we use in the algorithm.

Definition 29. Given $\alpha \in D$ we define the approximation $N_+^{\Theta(w)}\alpha$ of $N_+\alpha$ by

$$N_+^{\Theta(w)}\alpha = \frac{1}{16}\left(\left(\widetilde{|A_{0,0}|} \odot \widetilde{\sqrt{2}} \oplus \widetilde{|A_{0,1}|}\right)^{\odot(2)} \oplus \left(\widetilde{|A_{1,0}|} \odot \widetilde{\sqrt{2}} \oplus \widetilde{|A_{1,1}|}\right)^{\odot(2)}\right)^{\odot(2)} ,$$

where we assume that $w \geq 4$.

Lemma 30. *When w is constant, $N_+^\Theta \alpha$ can be computed in time $O(\log n)$, where n is the bitsize of $\alpha \in D$.*

Proof. The lemma is true, since to compute N_+^Θ we need only keep track of the number of trailing zeros in each number and perform constant w-precision arithmetic. □

Lemma 31. *Let $\alpha \in D$ and $w > 3$. Then*

$$\frac{1}{\Theta^6} N_+ \alpha \leq N_+^\Theta \alpha \leq N_+ \alpha$$

Proof. The right inequality is obvious. The left inequality follows by repeated application of Lemma 28.

$$\begin{aligned}
N_+\alpha &= \frac{1}{16}((|A_{0,0}|\sqrt{2} + |A_{0,1}|)^2 + (|A_{1,0}|\sqrt{2} + |A_{1,1}|)^2)^2 \\
&\leq \frac{1}{16}(\Theta^6(\widetilde{|A_{0,0}|} \odot \widetilde{\sqrt{2}} + \widetilde{|A_{0,1}|})^2 + \Theta^6(\widetilde{|A_{1,0}|} \odot \widetilde{\sqrt{2}} + \widetilde{|A_{1,1}|})^2)^2 \\
&\leq \frac{1}{16}(\Theta^{13}(\widetilde{|A_{0,0}|} \odot \widetilde{\sqrt{2}} \oplus \widetilde{|A_{0,1}|})^{\odot(2)} + \Theta^{13}(\widetilde{|A_{1,0}|} \odot \widetilde{\sqrt{2}} \oplus \widetilde{|A_{1,1}|})^{\odot(2)})^2 \\
&\leq \Theta^{30} N_+^\Theta \alpha
\end{aligned}$$

□

Recall that $\Theta = 1 + 2^{1-w}$. Thus we can make Θ^{30} arbitrarily close to 1 by choosing a sufficiently large w. This concludes the proof of Lemma 20, and explains our abuse of notation when writing N_+^Γ to denote the approximation within a factor $0 < \Gamma < 1$. A value of $w = 16$ gives a sufficiently good approximation for all practical purposes.