ON THE TAME KERNELS OF IMAGINARY CYCLIC QUARTIC FIELDS WITH CLASS NUMBER ONE

ZHANG LONG AND XU KEJIAN

Abstract. Tate first proposed a method to determine $K_2 \mathcal{O}_F$, the tame kernel of $F$, and gave the concrete computations for some special quadratic fields with small discriminant. After that, many examples for quadratic fields with larger discriminants are given, and similar works also have been done for cubic fields and for some special quartic fields with discriminants not large.

In the present paper, we investigate the case of more general imaginary cyclic quartic field $F = \mathbb{Q}(\sqrt{-D + B\sqrt{D}})$ with class number one and large discriminants. The key problem is how to decrease the huge theoretical bound appearing in the computation to a manageable one and the main difficulty is how to deal with the large-scale data emerged in the process of computation.

To solve this problem we have established a general architecture for the computation, in particular we have done the works: (1) the PARI’s functions are invoked in C++ codes; (2) the parallel programming approach is used in C++ codes; (3) in the design of algorithms and codes, the object-oriented viewpoint is used, so an extensible program is obtained.

As an application of our program, we prove that $K_2 \mathcal{O}_F$ is trivial in the following three cases: $B = 1, D = 2$ or $B = 2, D = 13$ or $B = 2, D = 29$.

In the last case, the discriminant of $F$ is 24389, hence, we can claim that our architecture also works for the computation of the tame kernel of a number field with discriminant less than 25000.

1. Introduction

Let $F$ be a number field and $\mathcal{O}_F$ the ring of algebraic integers of $F$, and let $K_2 \mathcal{O}_F$ denote the $K_2$ of $\mathcal{O}_F$. Garland [10] proved that $K_2 \mathcal{O}_F$ is a finite abelian group. However, $K_2 \mathcal{O}_F$ can be regarded as tame kernel.

In fact, let $K_2 F$ be the Milnor $K_2$-group, and let $k_v = \mathcal{O}_F / \mathcal{P}_v$ and $k_v^*$ the multiplicative group of $k_v$, where $\mathcal{P}_v$ is the prime ideal corresponding to a finite prime place $v$. Then we have the well-known tame homomorphism:

$$\partial_v : K_2 F \to k_v^*$$

which is defined by

$$\partial_v(\{x, y\}) = (-1)^{v(x)V(y)}\frac{x^{v(y)}y^{v(x)}}{y^{v(x)}x^{v(y)}}(\text{mod } \mathcal{P}_v),$$

2010 Mathematics Subject Classification. Primary 19C99, 19F15.

Key words and phrases. tame kernel, cyclic quartic field, multi-threaded parallel computing, Object-Oriented Program.
where \( v(x), v(y) \) denote the valuations of \( x, y \) with respect to the prime \( v \) respectively, and thus we have

\[
\partial = \bigoplus_v \partial_v : K_2 F \to \bigoplus_v k_v^*,
\]

where \( v \) runs over all finite places. The kernel \( \ker \partial \) is called the tame kernel of the field \( F \). D. Quillen [19] proved that \( \ker \partial = K_2 \mathcal{O}_F \).

There is no an effective algorithm for determining the tame kernel of a given number field directly, because it is defined noneffectively. The first method of determining the tame kernel of a given number field was proposed by J. Tate [21].

Now, we describe Tate’s method in more details.

Let \( N_v \) be the number \(|k_v|\), which is called the norm of \( v \), and let \( v_1, v_2, v_3, \ldots \) be all finite places of \( F \) ordered in such a way that \( N_{v_1} \leq N_{v_i+1} \), for \( i = 1, 2, 3, \ldots \).

Let \( S_m = \{v_1, \ldots, v_m\} \) \((S_0 = \emptyset)\), and let

\[
\mathcal{O}_m = \{a \in F : v(a) \geq 0, v \notin S_m\},
\]

\[
U_m = \{a \in F : v(a) = 0, v \notin S_m\}.
\]

Thus \( \mathcal{O}_0 \) and \( U_0 \) are just the ring of algebraic integers and the group of units respectively.

Let \( K_2^{S_m} F \) be the subgroup of \( K_2 F \) generated by symbols \( \{x, y\} \), where \( x, y \in U_m \). Then we have \( K_2 F = \bigcup_{m=1}^{\infty} K_2^{S_m} F \). Clearly, \( \partial_{v_m} \) induces the homomorphism

\[
\partial_{v_m} : \frac{K_2^{S_m} F}{K_2^{S_{m-1}} F} \to k_{v_m}^*.
\]

Bass and Tate [1] proved that for sufficiently large \( m \), \( \partial_{v_m} \) is isomorphic, which implies

\[
K_2 \mathcal{O}_F = \ker \left( \partial : K_2^{S_m} F \to \prod_{v \in S_m} k_v^* \right).
\]

Thus, if we can make the large \( m \) as small as possible and get sufficiently many relations satisfied by elements of \( K_2^{S_m} F \), then we may determine the tame kernel \( K_2 \mathcal{O}_F \). So the problem is reduced to finding conditions for \( \partial_{v_m} \) to be isomorphic for sufficiently large \( m \). The conditions were found by Tate.

Assume that the prime ideal \( \mathcal{P}_m \) of \( \mathcal{O}_m \) corresponding to \( v_m \) is generated by \( \pi_m \).

Define the morphisms:

\[
\alpha : U_m \to \frac{K_2^{S_m} F}{K_2^{S_{m-1}} F}, \quad \alpha(u) = \{u, \pi_m\} \mod K_2^{S_{m-1}} F,
\]

\[
\beta : U_m \to k_{v_m}^*, \quad \beta(u) = u \mod \pi_m.
\]

Then the conditions found by Tate are presented in the following theorem.

**Theorem 1.1.** [21] Suppose that prime ideal \( \mathcal{P} \) corresponding to a finite place \( v \notin S_m \) is generated by \( \pi \in \mathcal{O}_F \) and that \( U'_1 \) is a group generated by \( (1 + \pi U_m) \cap U_m \).

If there are subsets \( W_m, C_m, G_m \) of \( U_m \) satisfying the following conditions:

1. \( W_m \subseteq C_m U'_1 \) and \( U_m \) is generated by \( W_m \),
2. \( C_m G_m \subseteq C_m U'_1 \) and \( k_v^* \) is generated by \( \beta(G_m) \),
3. \( 1 \in C_m \cap \ker \beta \subseteq U'_1 \),

then \( \partial_v \) is an isomorphism.
Hence, according to Tate’s above method, to determine the tame kernel of a given number field, it suffices to construct suitable subsets $W_m, C_m, G_m$ of $U_m$ and determine the bound of $m$.

Using his method, Tate could give the analysis for the six first imaginary quadratic cases because in these cases the bound of $m$ is very small. More precisely, let $F = \mathbb{Q}(\sqrt{-d})$. Then Tate proved that $K_2 \mathcal{O}_F$ is trivial if $d = 1, 2, 3, 11$, and $K_2 \mathcal{O}_F \cong \mathbb{Z}/2\mathbb{Z}$ if $d = 7, 15$.

Subsequently, Qin [17, 18] investigated the cases $d = 6$ and 35 with a modification of the choice of the subset $C_m$ in Tate’s method, and nearly at the same time, Skalba [20] gave the computations of the cases $d = 5$ and 19 with the help of his generalized Thue theorem (GTT); essentially, it is also a modification of the choice of $C_m$. After that, for quadratic fields Browkin improved Skalba’s method to get a more accurate bound of $m$, which allowed him to compute the cases $d = 23$ and 31 [4, 5]. It should be pointed out that all of these works were done by hand.

The further computations for quadratic fields are due to Belabas and Gangl who used computers and determined the tame kernel for all $d$ up to 1000 with only 7 exceptions. [2]

The tame kernels of cubic fields had been investigated by Browkin in [7]. His numerical computations were performed using the package PARI/GP.

The cases of quartic fields are more complicated. Using Skalba’s GTT, Guo proved that $K_2 \mathcal{O}_F$ is trivial when $F = \mathbb{Q}(\zeta_8)$ (see [12]). He did it also by hand. When $F = \mathbb{Q}(\zeta_5) = \mathbb{Q}(\sqrt{-(5 + 2\sqrt{5}))}$, under the assumption of the Lichtenbaum conjecture, Browkin once conjectured in the paper [6] that the tame kernel $K_2 \mathcal{O}_F$ is trivial. In a recent paper, we confirmed Browkin’s conjecture [23]. However, the arithmetic properties of field $\mathbb{Q}(\zeta_5)$ are much more complicated than those of quadratic fields and biquadratic fields. Therefore the discussion is longer, and more cases are considered. Actually, we have to use PARI/GP and some other algorithms.

For further computations, the bound $m$ should be determined theoretically. This was solved by R. Groenewegen [11], who gave a theoretical bound of $m$ for a general number field. In this paper, for the cyclic quartic field we also find a way to obtain the theoretical bound, and in some cases our bounds are better than Groenewegen’s (Remark 4.7).

Thus, for a given number field, if the theoretical bound is good enough, that is, if it is a manageable, in another words, if the computation can be done by hand, then through constructing enough relations, we may determine the tame kernel of the given number field. But unfortunately, actually these theoretical bounds may be very large, far from being manageable. This weak point makes the concrete computation nearly impossible for a higher degree number field, even for a cyclic quartic field.

Hence, a new problem arises:

**Problem:** Whether one can give a practical method to decrease the theoretical bound to a manageable one?

Belabas and Gangl [2] considered this problem. In order to get a manageable bound of $m$, they proceed as follows. Let $T = S \cup \{v\}$ and assume that $K_2 \mathcal{O}_F \subseteq K_T^2 F$. They want to prove that, in fact, we already have $K_2 \mathcal{O}_F \subseteq K_S^2 F$. This will be used in the following situation: starting from the initial $S$ determined by the theoretical bound, we iterate this process, successively truncating $S$ by deleting its last element with respect to the given ordering, hoping to reduce the set of places to
a manageable size. This is a very natural way to decrease the theoretical bound to a manageable one, which has been used by many authors. But again unfortunately its concrete realization is not easy in general.

In fact, if the discriminant of the given number field is not large, then the difference between the theoretical bound and the manageable one is not large either, so it is easy to for one to do the work by writing a simple program or computing manually; but, if the discriminant is large, then the difference between the theoretical bound and the manageable one is also large, so the work-load must increase exponentially, as Balabas told us in a private letter, hence, in this case, we must face some challenges coming from dealing with the computation-intense task in the process solving the complex question.

In order to realize their plan, in particular in the construction of the set $C$, which is one of the most difficulties to overcome, Balabas and Gangl [2], use the following three algorithms: a) Fincke and Pohst’s algorithm; b) Method of lattice; c) LLL-algorithm. Balabas and Gangl’s plan was eventually adapted for arbitrary number fields and implemented in the PARI/GP scripting language, but so far, as they pointed out [2], parts of the program remain specific to the imaginary quadratic case.

In the present paper, we give a completely different and new approach. The key idea is that we use Object-Oriented Programming (OOP) and the Multi-threaded Parallel Technology.

It is well known that the idea of Object-Oriented Programming (OOP), developed as the dominant programming methodology in the early and mid 1990s, is to design data forms that correspond to the essential features of a problem. So OOP brings a new approach to the challenge of the large-scale programming. [16]

In this paper, to compute the tame kernels of imaginary cyclic quartic fields of class number one with large discriminant, with the object-oriented viewpoint we develop a program, the software framework of which is extensible and reusable and can be made as a base on which more tame kernels of number fields can be computed. Moreover, to visualize the program’s architectural blueprint, we also use Unified Modeling Language (UML) [?], which is a general-purpose, developmental and modeling language in the field of software engineering.

More precisely, in order to establish the software framework and visualize the architectural blueprint, we need to do the following works. Firstly, we need to reduce Tate’s theorem to a software engineering version, so as to give a main use case of a user’s interaction with the system. The use case is not only a beginning of building the software framework but also a main driving force. In order to visualize the use case, we give the use case diagram (see Figure 1), which can be regarded as a UML description of Tate’s theorem.

Secondly, we design three classes: $CquarField$, $Cideal$ and $Ccheck$ as the structure of our program since in the view of OOP all classes of a software constitute the core of the software framework. Moreover, using UML we represent the relationships with the classes as a static class diagram, which is generally used for general conceptual modelling of the systematic of a program and for detailed modelling translating the models into programming code. The relationships with the three classes are represented the following static class diagram (see Figure 2) and the detailed design of the three classes and the static class diagram are introduced in (4.3).
ON THE TAME KERNELS OF IMAGINARY CYCLIC QUARTIC FIELDS WITH CLASS NUMBER ONE

the system of computing the tame kernel

obtain the field

obtain the ideal

obtain C

obtain W

obtain G

check Con I(II)

Figure 1. the use case diagram

| CquarField | Ccheck | Cideal |
|------------|--------|--------|
|            | qf:CquarField | qf:CquarField |
|            | Ccheck(a:int, b:int, c:int, d:int) | qf:CquarField |
|            | Pr_chChndOn(n_thrd:int):bool | getSetInitW():GEN |
|            | Pr_chChndTw(n_thrd:int):bool | getSetInitG():GEN |
|            | Para_getSetInitC():void | Para_getSetInitC():void |

Figure 2. the static class diagram
Finally, in the view of software engineering, it is not enough to provide the use case diagram and the static class diagram to represent the program’s architecture, in another words, we must show how objects operate with one another and in what order. Since it is well known that in UML, a sequence diagram, which is an interaction diagram and also a construct of a message sequence chart, shows object interactions arranged in time sequence, thus we design the sequence diagram (see Figure 3) according to the relationships with the objects, which are represented in Tate’s theorem, and in view of the difficulties we must face during construction of the program of computing the tame kernel of imaginary cyclic quartic field, such as large-scale computing.

This is what we have done in this paper in the design of the program framework and the program architecture in UML. However, during building the program, we meet two difficulties.

One difficulty is how to create the codes which can be used to compute invariants of a number field. Though some authors have designed some excellent algorithms for the computation, the workload is so burdensome that it is almost impossible to implement so much algorithms for the computation of tame kernels. So it may be the viable option to use the third-party libraries to obtain the invariants. Hence, PARI library, looked on as a reliable component, provides a powerful support to our program.

The other difficulty is how to deal with the large-scale data emerging in the process of computation. In this study, we find that the amount of computation of tame kernels grows explosively as the discriminant and degree of extension of number fields get larger. In [2] Belabas and Gangl have computed some tame kernels of the quartic fields with absolute values of discriminants not large and the workloads in computation of the tame kernels of these quartic fields are nearly equal to that of $F = \mathbb{Q}(\zeta_5)$. But now, as an example, we compute the tame kernel of $F = \mathbb{Q}(\sqrt[5]{-(13+2\sqrt{13})})$ whose discriminant is 2917, and we find that the workloads for the computation of the tame kernels of $F = \mathbb{Q}(\zeta_5)$ and $F = \mathbb{Q}(\sqrt[5]{-(13+2\sqrt{13})})$ are not to be mentioned in the same breath. In fact, in the case of $F = \mathbb{Q}(\sqrt[5]{-(13+2\sqrt{13})})$, we once wrote some script codes with PARI/GP to compute its tame kernel. After deploying the codes on PC and running about 24 hours, we make a rough estimate of running time. It needs at least one year! So these script codes are not time-base.

Thus, it is for this reason that motivates us to design, in order to decrease the running time, the above architecture, which is an extensible, reusable and component-based application by associating the Multi-threaded Parallel Technology and PARI library with the implemented architecture. And at last, deploying the application and running about 2 hours, we obtain the result of tame kernel of $F = \mathbb{Q}(\sqrt[5]{-(13+2\sqrt{13})})$.

After that, we took about 3 months to compute the tame kernel of the number field $F = \mathbb{Q}(\sqrt[5]{-(29+2\sqrt{29})})$ whose discriminant is 24389.

In a private letter, Balabas told us that it took about 8 hours to obtain the tame kernel of $F = \mathbb{Q}(\sqrt[5]{-(13+2\sqrt{13})})$ by a program implementing the algorithms in the paper [2]. The program has been published in https://www.math.u-bordeaux.fr/~kbelabas/research/software.
ON THE TAME KERNELS OF IMAGINARY CYCLIC QUARTIC FIELDS WITH CLASS NUMBER ONE

Figure 3. the sequence diagram
We also tried ever to use the same program to compute the tame kernel of \( F = \mathbb{Q}(\sqrt{-29 + 2\sqrt{29}}) \). But, after running the program about 2 hours, a bug emerged and the program was interrupted. This story implies that although some kind of problems can be solved efficiently by using the existing program without difficulty, the computation of large-scale problems may be a nontrivial task, even a long-time running being acceptable, because of the restriction of the memory and CPU limitation. Therefore, the design of programs as well as its efficiency and reasonability may be essentially depended on the scale of computation.

Hence, as an application of our program, now we are sure from the above computation that our architecture also works for the computation of the tame kernel of a number field with discriminant less than 25000.

In particular, as concrete examples, we have proved the following theorem.

**Theorem 1.2.** Let \( F = \mathbb{Q}(\sqrt{-29 + 2\sqrt{29}}) \) be cyclic quartic field. Then the tame kernel \( K_2^F \) is trivial in the following cases:

(i) \( B = 2, D = 5 \), i.e., \( F = \mathbb{Q}(\sqrt[5]{5}) \);
(ii) \( B = 1, D = 2 \);
(iii) \( B = 2, D = 13 \);
(iv) \( B = 2, D = 29 \).

**Remark 1.3.** (i) We have \( h_F = 1 \) in the four cases in Lemma 2.1.

(ii) By the present algorithms, the computation of the tame kernel of \( \mathbb{Q}(\sqrt[5]{5}) \) is quite easy.

In the following, the conditions \( W_m \subseteq C_m U'_1 \) and \( C_m G_m \subseteq C_m U'_1 \) in Theorem 1.1 will be referred to be condition I and condition II respectively.

2. The cyclic quartic fields

The following explicit representation of a cyclic quartic field is proved in the reference [13].

**Lemma 2.1.** If \( F \) is a real or imaginary cyclic quartic extension of \( \mathbb{Q} \), then there are integers \( A, B, C \) and \( D \) such that

\[
F = \mathbb{Q}\left(\sqrt{A(D + B\sqrt{D})}\right) = \mathbb{Q}\left(\sqrt{A(D - B\sqrt{D})}\right)
\]

where

\[
\begin{align*}
A & \text{ is squarefree and odd}, \\
D & = B^2 + C^2 \text{ is squarefree, } B > 0, \ C > 0, \\
A \text{ and } D & \text{ are relatively prime.}
\end{align*}
\]

Moreover, any field satisfying (2.1) and (2.2) is cyclic quartic extension of \( \mathbb{Q} \), and the representation of \( F \) is unique in the sense that if we have another representation, say \( F = \mathbb{Q}(\sqrt{A_1(D_1 + B_1\sqrt{D_1})}) \), where \( A_1, B_1, C_1 \) and \( D_1 \) are integers satisfying the conditions of (2), then \( A = A_1, B = B_1, C = C_1 \) and \( D = D_1 \).

On the other hand, it is given in the reference [13] a table of all the imaginary cyclic quartic fields \( F = \mathbb{Q}\left(\sqrt{A(D + B\sqrt{D})}\right) \), where \( A, B, C \) and \( D \) are integers...
satisfying the condition (2.2). Now, we can list all imaginary cyclic quartic fields with class number one as follows.

Case 1: \( F = \mathbb{Q}\left(\sqrt{-5 + 2\sqrt{5}}\right) \), where \( A = -1, B = 2, C = 1, D = 5 \);

Case 2: \( F = \mathbb{Q}\left(\sqrt{-13 + 2\sqrt{13}}\right) \), where \( A = -1, B = 2, C = 3, D = 13 \);

Case 3: \( F = \mathbb{Q}\left(\sqrt{-2 + \sqrt{2}}\right) \), where \( A = -1, B = 1, C = 1, D = 2 \);

Case 4: \( F = \mathbb{Q}\left(\sqrt{-29 + 2\sqrt{29}}\right) \), where \( A = -1, B = 2, C = 5, D = 29 \);

Case 5: \( F = \mathbb{Q}\left(\sqrt{-37 + 6\sqrt{37}}\right) \), where \( A = -1, B = 6, C = 1, D = 37 \);

Case 6: \( F = \mathbb{Q}\left(\sqrt{-53 + 2\sqrt{53}}\right) \), where \( A = -1, B = 2, C = 7, D = 53 \);

Case 7: \( F = \mathbb{Q}\left(\sqrt{-61 + 6\sqrt{61}}\right) \), where \( A = -1, B = 6, C = 5, D = 61 \);

In [14], the integral basis of the cyclic quartic field \( F = \mathbb{Q}\left(\sqrt{A(D + B\sqrt{D})}\right) \) is given as follows.

**Lemma 2.2.** Let \( F = \mathbb{Q}\left(\sqrt{A(D + B\sqrt{D})}\right) \) be a cyclic quartic extension of \( \mathbb{Q} \), where \( A, B, C \) and \( D \) satisfy the condition (2.2) in Lemma 2.1. Set

\[
a' = \sqrt{A(D + B\sqrt{D})}, \quad b' = \sqrt{A(D - B\sqrt{D})};
\]

Then an integral basis for \( F \) is given as follows.

(i) \( \{1, \sqrt{D}, a', b'\} \) if \( D \equiv 0(\text{mod} 2) \);

(ii) \( \{1, \frac{1}{2}(1 + \sqrt{D}), a', b'\} \) if \( D \equiv B \equiv 1(\text{mod} 2) \);

(iii) \( \left\{1, \frac{1}{2}(1 + \sqrt{D}), \frac{1}{2}(a' + b'), \frac{1}{2}(a' - b')\right\} \)
    if \( D \equiv 1(\text{mod} 2), B \equiv 0(\text{mod} 2), A + B \equiv 3(\text{mod} 4) \);

(iv) \( \left\{1, \frac{1}{2}(1 + \sqrt{D}), \frac{1}{4}(1 + \sqrt{D} + a' + b'), \frac{1}{4}(1 - \sqrt{D} + a' - b')\right\} \)
    if \( D \equiv 1(\text{mod} 2), B \equiv 0(\text{mod} 2), A + B \equiv 1(\text{mod} 4), A \equiv C(\text{mod} 4) \);

(v) \( \left\{1, \frac{1}{2}(1 + \sqrt{D}), \frac{1}{4}(1 + \sqrt{D} + a' - b'), \frac{1}{4}(1 - \sqrt{D} + a' + b')\right\} \)
    if \( D \equiv 1(\text{mod} 2), B \equiv 0(\text{mod} 2), A + B \equiv 1(\text{mod} 4), A \equiv -C(\text{mod} 4) \);

Hence, the integral bases of Case 2, of Case 3, Case 4, Case 5, Case 7 and of Case 6 are respectively

\[
\left\{1, \sqrt{D}, a', b'\right\};
\]
\[
\left\{1, \frac{1}{2}(1 + \sqrt{D}), \frac{1}{4}(1 + \sqrt{D} + a' - b'), \frac{1}{4}(1 - \sqrt{D} + a' + b')\right\};
\]
\[
\left\{1, \frac{1}{2}(1 + \sqrt{D}), \frac{1}{4}(1 + \sqrt{D} + a' + b'), \frac{1}{4}(1 - \sqrt{D} + a' - b')\right\}.
\]
Lemma 2.3. Let \( F = \mathbb{Q}\left( \sqrt{-D + B\sqrt{D}} \right) \) be a cyclic quartic extension of \( \mathbb{Q} \) with class number \( h(F) = 1 \), where \( B, C \) and \( D \) satisfy the condition (2.2) in Lemma 2.1. Set \( \beta = i\sqrt{D + B\sqrt{D}} \) and \( F = \mathbb{Q}(\beta) \). Then the following statements hold.

(i) The minimal polynomial of \( \beta \) over \( \mathbb{Q} \) is

\[
f(x) = x^4 + 2Dx^2 + (D^2 - DB^2).
\]

(ii) The four conjugated roots of \( \beta \) are

\[
\beta_1 = \beta = ia, \quad \beta_2 = \bar{\beta} = -ia, \quad \beta_3 = ib, \quad \beta_4 = -ib,
\]

where \( a = \sqrt{D + B\sqrt{D}} \) and \( b = \sqrt{D - B\sqrt{D}} \).

(iii) The Galois group \( \text{Gal}(F/\mathbb{Q}) \) equals \( \langle \sigma \rangle \) with \( \sigma \) satisfying

\[
\sigma(\beta_1) = \beta_4, \quad \sigma(\beta_2) = \beta_3, \quad \sigma(\beta_3) = \beta_1, \quad \sigma(\beta_4) = \beta_2.
\]

(iv) The rank \( r(U) \) of unit group \( U \) of \( F \) is 1. We denote the fundamental unit by \( \xi \).

(v) In Case 1, Case 3, Case 4, Case 5, Case 7, the field \( F \) has the same integral base, which is

\[
\gamma_0 = 1, \quad \gamma_1 = \frac{1}{2}(1 + \sqrt{D}), \quad \gamma_2 = \frac{1}{4}(1 + \sqrt{D} + \beta - \beta_3), \quad \gamma_3 = \frac{1}{4}(1 - \sqrt{D} + \beta + \beta_3).
\]

Moreover, the transition matrix from \( 1, \beta, \beta^2, \beta^3 \) to \( \gamma_0, \gamma_1, \gamma_2, \gamma_3 \) is

\[
M_1 = \begin{pmatrix}
1 & 0 & 0 & 0 \\
\frac{B-D}{2B} & \frac{B-D}{4B} & \frac{BC-B^2-D}{14B} & -\frac{1}{4B} \\
\frac{B+D}{4B} & \frac{BC+B^2+D}{4B} & \frac{1}{4B} & -\frac{1}{4B}
\end{pmatrix}
\]

(vi) In Case 2 and Case 6, the field \( F \) has the same integral base, which is

\[
\gamma_0 = 1, \quad \gamma_1 = \frac{1}{2}(1 + \sqrt{D}), \quad \gamma_2 = \frac{1}{4}(1 + \sqrt{D} + \beta + \beta_3), \quad \gamma_3 = \frac{1}{4}(1 - \sqrt{D} + \beta - \beta_3).
\]

Moreover, the transition matrix from \( 1, \beta, \beta^2, \beta^3 \) to \( \gamma_0, \gamma_1, \gamma_2, \gamma_3 \) is

\[
M_2 = \begin{pmatrix}
1 & 0 & 0 & 0 \\
\frac{B-D}{2B} & \frac{B-D}{4B} & \frac{BC-B^2-D}{14B} & -\frac{1}{4B} \\
\frac{B+D}{4B} & \frac{BC+B^2+D}{4B} & \frac{1}{4B} & -\frac{1}{4B}
\end{pmatrix}
\]

Proof. The proofs of (i),(ii),(iii) and (iv) are easy. So we only prove (v) and (vi).

We will express \( \beta_3 = ib \) by \( 1, \beta, \beta^2, \beta^3 \). Assume that

\[
\beta_3 = ib = x_0 + x_1ia + x_2(ia)^2 + x_3(ia)^3,
\]

where \( a = \sqrt{D + B\sqrt{D}}, \ b = \sqrt{D - B\sqrt{D}} \) and \( x_0, x_1, x_2, x_3 \in \mathbb{Q} \). Then the following equations hold:

\[
x_0 - a^2x_2 = 0
\]

\[
b - ax_1 + a^3x_3 = 0
\]
From (2.6), we have $x_0 = x_2 = 0$. However, from (2.7), we get
\[
b^2 = a^2(x_1 - a^2 x_3)^2 \\
= a^2(x_1^2 + a^2 x_3^2 - 2a^2 x_1 x_3) \\
= (D + B \sqrt{D})[x_1^2 + (D + B \sqrt{D})x_3^2 - 2(D + B \sqrt{D})x_1 x_3] \\
= (D + B \sqrt{D})[x_1^2 + (D^2 + B^2)x_3^2 - 2Dx_1 x_3 + (2BDx_3^2 - 2Bx_1 x_3)\sqrt{D}] \\
= [Dx_1^2 + (D^3 + 3B^2 D^2)x_3^2 - 2(D^2 + B^2)Dx_1 x_3] \\
+ [Bx_1^2 + (3BD^2 + B^3)Dx_3^2 - 4BDx_1 x_3]\sqrt{D}.
\]

By comparing with the both sides of the equality, we can get the system of equations on $x_1$ and $x_3$
\[
\begin{align*}
(2.8) & \\
D &= Dx_1^2 + (D^3 + 3B^2 D^2)x_3^2 - 2(D^2 + B^2)Dx_1 x_3 \\
(2.9) & \\
-B &= Bx_1^2 + (3BD^2 + B^3)Dx_3^2 - 4BDx_1 x_3. \\
i.e. \quad & \\
(2.10) & \\
1 &= x_1^2 + (D^2 + 3B^2)Dx_3^2 - 2(D + B^2)x_1 x_3 \\
(2.11) & \\
-1 &= x_1^2 + (3D^2 + B^2)Dx_3^2 - 4Dx_1 x_3.
\end{align*}
\]

Adding the two equations, we have
\[
(2.12) \quad x_1^2 + (2D^2 + 2B^2)Dx_3^2 - (3D + B^2)x_1 x_3 = 0.
\]

If $x_3 = 0$, clearly we have $x_1 = 0$, impossible. Thus $\frac{x_1}{x_3}$ is a root of the equation:
\[
(2.13) \quad x^2 - (3D + B^2)x + 2D(D + B^2) = 0.
\]

Clearly, $2D, D + B^2$ are the two roots of (2.13), so
\[
\frac{x_1}{x_3} = 2D \quad \text{or} \quad \frac{x_1}{x_3} = D + B^2.
\]

If $x_1 = 2Dx_3$, then from (2.10), we can get that $(D^2 - B^2)Dx_3^2 = 1$. So we have
\[
x_3 = \pm \frac{\sqrt{D}}{CD}, \quad x_1 = \pm 2\sqrt{D}.
\]

However, putting these expressions in (2.11), we get immediately a contradiction. Hence, we must have $x_1 = (D + B^2)x_3$. Therefore from (2.11) we get
\[
-1 = (B^2 + D^2)x_3^3 + (3D^2 + 2B^2)Dx_3^2 - 4D(D + B^2)x_3^2 = -B^2C^2 x_3^2.
\]

Thus $x_3 = \pm \frac{1}{BC}$. We can check that $x_0 = x_2 = 0, x_1 = \frac{D + B^2}{BC}$ and $x_3 = \frac{1}{BC}$ satisfy the equation (2.5), which means
\[
\beta_3 = \frac{D + B^2}{BC} \cdot \beta + \frac{1}{BC} \cdot \beta^3.
\]

Note that $\sqrt{D} = \frac{D}{B} - \frac{\beta}{B}$. Then, in Case 1, Case 3, Case 4, Case 5, Case 7, we can express $\gamma_0, \gamma_1, \gamma_2, \gamma_3$ by $1, \beta, \beta^2, \beta^3$ as follows.
\[
\begin{pmatrix}
\gamma_0 \\
\gamma_1 \\
\gamma_2 \\
\gamma_3
\end{pmatrix} = \begin{pmatrix}
\frac{1}{BD} & 0 & 0 & 0 \\
\frac{B + D}{4B} & \frac{BC + B^2 - D}{4BC} & -\frac{1}{2B} & 0 \\
\frac{B + D}{4B} & \frac{B^2 - B^2 - D}{4BC} & -\frac{1}{4B} & -\frac{1}{4BC} \\
\frac{B + D}{4B} & \frac{B^2 + B^2 + D}{4BC} & \frac{1}{4B} & \frac{1}{4BC}
\end{pmatrix} \times M_1 \begin{pmatrix}
1 \\
\beta \\
\beta^2 \\
\beta^3
\end{pmatrix}.
\]
Similarly, in Case 2 and Case 6, we get

\[
\begin{pmatrix}
\gamma_0 \\
\gamma_1 \\
\gamma_2 \\
\gamma_3
\end{pmatrix} = \begin{pmatrix}
1 & -D & 0 & 0 \\
B & -D & BC & 0 \\
0 & BC^2 + B^2 & -B^2 & 0 \\
0 & 0 & BC - B^2 & -D
\end{pmatrix} \begin{pmatrix}
1 & \beta \\
\beta & \beta^2 \\
\beta^2 & \beta^3
\end{pmatrix} = M_2 \begin{pmatrix}
1 \\
\beta \\
\beta^2 \\
\beta^3
\end{pmatrix}.
\]

\[
\square
\]

3. The tame kernel of an imaginary cyclic quartic field

3.1. Lemmas.

Lemma 3.1. Let \( F = \mathbb{Q}(\sqrt{-D + B\sqrt{D}}) \) be a cyclic quartic field with the class number \( h(F) = 1 \) and let \( \beta = ia \) with \( a = \sqrt{D + B\sqrt{D}} \). Then, for any prime ideal \( \mathcal{P} \) of \( F \), there exists an element \( \alpha \in \mathcal{O}_F \) satisfying

(i) \( \mathcal{P} = \langle \alpha \rangle \);

(ii) \(|\sigma(\xi)| \leq \left| \frac{\sigma(\alpha)}{\alpha} \right| \leq |\xi|\), where \( \xi \) is the fundament unit of \( F \). Moreover, we have

\[
\left| \frac{N(\alpha)|^\frac{1}{4}}{|\xi|^\frac{1}{4}} \leq |\alpha| \leq \left| \frac{N(\alpha)|^\frac{1}{4}}{|\sigma(\xi)|^\frac{1}{4}} \right|.
\]

Proof. Because the class number \( h_F = 1 \), the prime ideal \( \mathcal{P} \) of \( F \) is a principal ideal, i.e. \( \mathcal{P} = \langle y \rangle \) for some \( y \in \mathcal{O}_F \).

i). If \(|\sigma(\xi)| \leq \left| \frac{\sigma(y)}{y} \right| \leq |\xi|\), let \( \alpha = y \). Then the lemma is true.

ii). If \(|\frac{\sigma(y)}{y}| > |\xi|\), since \( \frac{\sigma(\xi)}{\xi} < 1 \), there is a positive integer \( k \) satisfying

\[
|\sigma(y)| \left| \frac{\sigma(\xi)}{\xi} \right|^k \leq |\xi| < |\sigma(y)| \left| \frac{\sigma(\xi)}{\xi} \right|^{k-1}.
\]

Let \( \alpha = y \xi^{k} \). Then, we get \( \left| \frac{\sigma(\alpha)}{\alpha} \right| \leq |\xi| \). However \( \left| \frac{\sigma(\alpha)}{\alpha} \right| = |\sigma(\xi)| \left| \frac{\sigma(y^{k-1} \xi^{k-1})}{y \xi^{k-1}} \right| > |\sigma(\xi)| |\xi| > |\sigma(\xi)| \).

iii). If \(|\frac{\sigma(y)}{y}| < |\sigma(\xi)|\), as in ii), there is a positive integer \( k \) such that

\[
|\sigma(y)| \left| \frac{\xi}{\sigma(\xi)} \right|^{k-1} < |\sigma(\xi)| \leq \left| \frac{\sigma(y)}{y} \right| \left| \frac{\xi}{\sigma(\xi)} \right|^k.
\]

Let \( \alpha = \frac{y}{\xi^k} \). Thus by (14), we have

\[
|\sigma(\xi)| \leq \left| \frac{\sigma(\alpha)}{\alpha} \right| \leq |\xi|.
\]

So

\[
|\sigma(\xi)| |\alpha|^2 \leq |\sigma(\alpha)||\alpha| \leq |\xi||\alpha|^2.
\]

Hence

\[
|\sigma(\xi)|^2 |\alpha|^4 \leq |N(\alpha)| \leq |\xi|^2 |\alpha|^4.
\]

Therefore

\[
\left| \frac{N(\alpha)|^\frac{1}{4}}{|\xi|^\frac{1}{4}} \right| \leq |\alpha| \leq \left| \frac{N(\alpha)|^\frac{1}{4}}{|\sigma(\xi)|^\frac{1}{4}} \right|.
\]
We denote \([t]\) to the nearest integer number to \(t\). Let \(\{t\} = t - [t]\). So \(\{t\} \in [-\frac{1}{2}, \frac{1}{2}]\).

**Lemma 3.2.** For any \(0 \neq \alpha, x \in \mathcal{O}_F\), there is a \(y \in \mathcal{O}_F\) such that

\[
|x - \alpha y| \leq c_1|\alpha|,
\]

\[
|\sigma(x - \alpha y)| \leq c_2|\sigma(\alpha)|,
\]

where \(c_1, c_2\) are constants depending only on the field \(F\), i.e., on \(A, B, C\) and \(D\). So

\[
N(x - \alpha y) \leq c_1^2 c_2^2 N(\alpha).
\]

**Proof.** Assume that \(\frac{x}{\alpha} = k_0 \gamma_0 + k_1 \gamma_1 + k_2 \gamma_2 + k_3 \gamma_3\) where \(\gamma_0, \gamma_1, \gamma_2, \gamma_3\) are the integral basis of \(F\) and \(k_i \in \mathbb{Q}, i = 0, 1, 2, 3\). Let

\[
y = [k_0] \gamma_0 + [k_1] \gamma_1 + [k_2] \gamma_2 + [k_3] \gamma_3 \in \mathcal{O}_F.
\]

We will show that \(y\) satisfies the requirement.

Suppose that

\[
z = x - y \alpha = \left(\sum_{i=0}^{3} k_i \gamma_i\right) \alpha - \left(\sum_{i=0}^{3} k_i \gamma_i\right) \alpha = \left(\sum_{i=0}^{3} k_i \gamma_i\right) \alpha = \left(\sum_{i=0}^{3} z_i \gamma_i\right) \alpha,
\]

where \(z_i = \{k_i\} \in [-\frac{1}{2}, \frac{1}{2}] \cap \mathbb{Q}\).

Let \(M = M_1\) or \(M_2\), and let \(z' = \sum_{i=0}^{3} z_i \gamma_i\). We can compute the maximal value of \(|z|\). Let \(g = |z|^2\). Then

\[
g = |z|^2 = |z'|^2|\alpha|^2
\]

\[
= (z_0, z_1, z_2, z_3) M \begin{pmatrix}
1 & -ia & -a^2 & -a^3 \\
i \alpha & a^2 & -ia^3 & -a^4 \\
-a^2 & ia^3 & a^4 & -ia^5 \\
-ia^3 & a^4 & ia^5 & a^6
\end{pmatrix} M^T \begin{pmatrix}
z_0 \\
z_1 \\
z_2 \\
z_3
\end{pmatrix} |\alpha|^2,
\]

where

\[
H_1 := \begin{pmatrix}
1 & 0 & -a^2 & 0 \\
0 & a^2 & 0 & -a^4 \\
-a^2 & 0 & a^4 & 0 \\
0 & -a^4 & 0 & a^6
\end{pmatrix}.
\]

Let

\[
h_1(z_0, z_1, z_2, z_3) := (z_0, z_1, z_2, z_3) M H_1 M^T \begin{pmatrix}
z_0 \\
z_1 \\
z_2 \\
z_3
\end{pmatrix}.
\]

By pari/gp, we can check that the values of \(h_1(z_0, z_1, z_2, z_3)\) on those stationary are zero. Thus \(h_1(z_0, z_1, z_2, z_3)\) reaches its maximal value on the boundary.

Hence, for any \(A, B, C\) and \(D\), we have

\[
|x - y \alpha| = |z| \leq |z'||\alpha| \leq c_1^2|\alpha|,
\]
where
\[ c'_1 = \max \{ h(z_0, z_1, z_2, z_3) : z_i = -\frac{1}{2} \text{ or } \frac{1}{2}, i = 0, 1, 2, 3 \}. \]

Similarly, let
\[ h_2(z_0, z_1, z_2, z_3) = (z_0, z_1, z_2, z_3)MH_2M^T \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix}. \]

with
\[ H_2 := \begin{pmatrix} 1 & 0 & -b^2 & 0 \\ 0 & b^2 & 0 & -b^4 \\ -b^2 & 0 & b^4 & 0 \\ 0 & -b^4 & 0 & b^6 \end{pmatrix}. \]

Then we have
\[ |\sigma(x - y\alpha)| = |\sigma(z)| \leq |\sigma(z')||\sigma(\alpha)| \leq c'_2 |\sigma(\alpha)|, \]

where
\[ c'_2 = \max \{ h_2(z_0, z_1, z_2, z_3) : z_i = -\frac{1}{2} \text{ or } \frac{1}{2}, i = 0, 1, 2, 3 \}. \]

However both
\[ |z'|^2 = (z_0, z_1, z_2, z_3)MH_1M^T \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} \]

and
\[ |\sigma(z')|^2 = (z_0, z_1, z_2, z_3)MH_2M^T \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} \]

are positive definite quadratic forms determined by \( a > 0 \) and \( b > 0 \). So \( |z'| \) and \( |\sigma(z')| \) reach maximal value at same point. Let \( c_i = c'_i^{\frac{1}{2}}, i = 1, 2 \). Then the proof is completed. \( \square \)

3.2. Construction of \( W_m, C_m, G_m \). Let \( F = \mathbb{Q}\left(\sqrt{-(D + B\sqrt{D})}\right) \) be a cyclic quartic field with the class number \( h(F) = 1 \), and let \( S_{m+1} = \{v_1, v_2, \ldots, v_{m+1}\} \), where \( v_i \) corresponds to the prime ideal \( \mathcal{P}_i := \mathcal{P}_{v_i} \) for \( i = 1, 2, \ldots, m + 1 \). In order to use Theorem 1.1 to compute the tame kernel \( K_2\mathcal{O}_F \), we construct \( W_m, C_m \) and \( G_m \) as follows.

Firstly, by Lemma 3.1, for each \( i \) there exists an \( \alpha_i \in \mathcal{O}_F \) satisfying \( \mathcal{P}_i : (\alpha_i) \) and \( |\sigma(\xi)| \leq \frac{|\sigma(\alpha_i)|}{\alpha_i} \leq |\xi| \), where \( i = 1, 2, \ldots, m + 1 \). Thus we define
\[ W_m = \{\alpha_1, \alpha_2, \ldots, \alpha_m\} \cup \{-1, \xi\}. \]

Clearly, from the construction of \( W_m \), we know immediately that \( U_m \) can be generated by \( W_m \).

Secondly, let
\[ C'_m = \{c \in \mathcal{O}_K : |c| \leq c_1 |\alpha_{m+1}|, |\sigma(c)| \leq c_2 |\sigma(\alpha_{m+1})|\}. \]
We will estimate the term $|c| = |\sigma(c)|$, which implies that condition (iii) of Theorem 1.1 is satisfied. In the following, we will prove that there must exist a $C_m$ which satisfies condition I and condition II further.

Finally, let $\delta := (\frac{2}{3})^\frac{2}{3} |D|\frac{1}{3}$ and define

$$G_m' = \left\{ g \in \mathcal{O}_K : |g| \leq \delta N(P_{m+1})\frac{1}{3}, |\sigma(g)| \leq \delta N(P_{m+1})\frac{1}{3} \right\}.$$

When $N(P_{m+1}) > \delta^8$, by GTT theorem and the proof of Lemma 1.2 in [10], there exists a subset $G_m \subseteq U_m$ with $G_m \subseteq G'_m$ such that $k^*_v$ can be generated by $\beta(G_m)$, which means the second part of condition (ii) in Theorem 1.1 is satisfied.

3.3. Theoretical bounds.

3.3.1. The bounds in imaginary cyclic quartic field case. The following lemma is very helpful.

**Lemma 3.3.** Suppose that the elements $a, b \in \mathcal{O}_F \cap U_m$ satisfy the conditions $a \equiv b (\text{mod} \ P_{m+1})$ and $N(a - b) < N^2(P_{m+1})$. Then $\frac{a}{b} \in U'_1$.

**Proof.** See Claim 2 in the proof of Lemma 3.4 in [12].

Define

$$c' = \max \left\{ c_1 \frac{|\sigma(\xi)|}{|\xi|} + c_2 \frac{|\xi|}{|\sigma(\xi)|}, c_2 \frac{|\sigma(\xi)|}{|\xi|} + c_1 \frac{|\xi|}{|\sigma(\xi)|} \right\},$$

**Lemma 3.4.** If $N(P_{m+1}) \geq \left(1 + c_1 c_2 + c'\right)^2$, then $W_m \subseteq C_m U'_1$, i.e., condition I is satisfied.

**Proof.** By Lemma 3.3, if for any $w \in W_m$ there always exists a $c \in C_m$ satisfying $c \equiv w (\text{mod} \ P_{m+1})$ and $N(w - c) < N^2(P_{m+1})$, then we have $W_m \subseteq C_m U'_1$. So it suffices to investigate when the inequality $N(w - c) < N^2(P_{m+1})$ holds.

However, we have

$$N(w - c) = |w - c||\sigma(w) - \sigma(c)||\sigma^2(w) - \sigma^2(c)||\sigma^3(w) - \sigma^3(c)|$$

$$= |w - c||w - \tilde{c}||\sigma(w) - \sigma(c)||\sigma^2(w) - \sigma^2(c)|$$

$$= (|w - c||\sigma(w) - \sigma(c)|)^2.$$

We will estimate the term $|w - c||\sigma(w) - \sigma(c)|$.

First we have

$$|w - c||\sigma(w) - \sigma(c)| = |w\sigma(w) - w\sigma(c) - c\sigma(w) + c\sigma(c)|$$

$$\leq |w\sigma(w)| + |w\sigma(c)| + |c\sigma(w)| + |c\sigma(c)|$$

$$= N^\frac{1}{4}(w) + |w\sigma(c)| + |c\sigma(w)| + N^\frac{1}{4}(c).$$

From the construction of $W_m$ and $C_m$, we have $N^\frac{1}{4}(w) \leq N^\frac{1}{4}(P_{m+1})$ and $N^\frac{1}{4}(c) \leq c_1 c_2 N^\frac{1}{4}(P_{m+1})$.

Now we estimate the term $|w\sigma(c)| + |c\sigma(w)| = |c\sigma(w)| + \frac{N^\frac{1}{4}(w)N^\frac{1}{4}(c)}{|c\sigma(w)|}$. By Lemma 3.1, for any $c \in C_m$ we have

$$|c| \leq c_1 |\alpha_{m+1}| \leq c_1 \frac{N^\frac{1}{4}(\alpha_{m+1})}{|\sigma(\xi)|^\frac{1}{4}}.$$
In virtue of \[ |\sigma(\alpha_{m+1})| \leq |\xi| \text{ and } |N(\alpha_{m+1})| = |\alpha_{m+1}|^2 |\sigma(\alpha_{m+1})|^2, \]
we have
\[ |c| = \left| \frac{N^\downarrow(c)}{|\sigma(c)|} \right| \geq \frac{N^\downarrow(c)}{c_2|\sigma(\alpha_{m+1})|} = \frac{N^\downarrow(c)|\alpha_{m+1}|}{c_2N^\downarrow(\alpha_{m+1})}. \]

\[ \geq \frac{N^\downarrow(c)N^\downarrow(\alpha_{m+1})}{c_2|\xi||N^\downarrow(\alpha_{m+1})|} = \frac{N^\downarrow(c)}{c_2|\xi|N^\downarrow(\alpha_{m+1})}. \]

So we have
\[ \frac{N^\downarrow(c)}{c_2|\xi|N^\downarrow(\alpha_{m+1})} \leq |c| \leq \frac{c_1N^\downarrow(\alpha_{m+1})}{|\sigma(\alpha)|}. \]

When \( w \neq \xi \in \mathcal{W}_m \), from the construction of \( \mathcal{W}_m \), we have
\[ \frac{|\sigma(\xi)||N^\downarrow(c)N^\downarrow(\alpha_{m+1})}{c_2|\xi|N^\downarrow(\alpha_{m+1})} \leq |c\sigma(w)| \leq \frac{|\xi||w|}{|\sigma(\alpha)|}N^\downarrow(w). \]

When \( w = -1 \) or \( \xi \), clearly the inequality above also holds. Thus we get
\[ \frac{|\sigma(\xi)||N^\downarrow(c)N^\downarrow(\alpha_{m+1})}{c_2|\xi|N^\downarrow(\alpha_{m+1})} \leq |c\sigma(w)| \leq \frac{c_1N^\downarrow(\alpha_{m+1})}{|\sigma(\alpha)|}. \]

It is easy to show the function \( f(x) = x + \frac{N^\downarrow(w)N^\downarrow(c)}{x} \) meet its maximal value on the boundary. However the function values of \( f(x) \) on boundary \( x = \frac{|\sigma(\xi)||N^\downarrow(c)N^\downarrow(\alpha_{m+1})}{c_2|\xi|N^\downarrow(\alpha_{m+1})} \) or \( c_1\frac{|\xi|}{|\sigma(\xi)|}N^\downarrow(\alpha_{m+1}) \) can be computed as follows,

\[ \frac{|\sigma(\xi)||N^\downarrow(c)N^\downarrow(\alpha_{m+1})}{c_2|\xi|N^\downarrow(\alpha_{m+1})} + \frac{c_2|\xi|}{|\sigma(\xi)|}N^\downarrow(w)N^\downarrow(P_{m+1}) \]
\[ \leq c_1\frac{|\sigma(\xi)||N^\downarrow(c)N^\downarrow(\alpha_{m+1})}{|\xi|N^\downarrow(\alpha_{m+1})} + \frac{c_2|\xi|}{|\sigma(\xi)|}N^\downarrow(\alpha_{m+1}) \]
\[ = \left( c_1\frac{|\sigma(\xi)|}{|\xi|} + c_2\frac{|\xi|}{|\sigma(\xi)|} \right)N^\downarrow(\alpha_{m+1}) \]

and

\[ \frac{c_1|\xi|}{|\sigma(\xi)|}N^\downarrow(w)N^\downarrow(P_{m+1}) + \frac{|\sigma(\xi)||N^\downarrow(c)N^\downarrow(\alpha_{m+1})}{c_1|\xi|N^\downarrow(\alpha_{m+1})} \]
\[ \leq c_1\frac{|\xi|}{|\sigma(\xi)|}N^\downarrow(\alpha_{m+1}) + c_2|\xi| \frac{N^\downarrow(c)N^\downarrow(\alpha_{m+1})}{N^\downarrow(\alpha_{m+1})} \]
\[ = \left( c_2\frac{|\sigma(\xi)|}{|\xi|} + c_1\frac{|\xi|}{|\sigma(\xi)|} \right)N^\downarrow(\alpha_{m+1}). \]

So we have
\[ |c\sigma(w)| + |w\sigma(c)| = |c\sigma(w)| + \frac{N^\downarrow(w)N^\downarrow(c)}{|c\sigma(w)|} \leq c'N^\downarrow(\alpha_{m+1}), \]
where $c' = \max\left\{ c_1 \frac{\|\sigma(c)\|}{\|c\|} + c_2 \frac{\|c\|}{\|\sigma(c)\|}, c_3 \frac{\|\sigma(c)\|}{\|c\|} + c_1 \frac{\|c\|}{\|\sigma(c)\|} \right\}$. Summarily we get

$$|N(w - c)| = |(w - c)\sigma(w) - \sigma(c)|^2$$

$$\leq (N^\frac{1}{2}(w) + |w\sigma(c)| + |c\sigma(w)| + N^\frac{1}{2}(c))^2$$

$$\leq \left(1 + c_1c_2 + c'\right)^2 N(\alpha_{m+1}).$$

So when $N(\alpha_{m+1}) = N(P_{m+1}) > \left(1 + c_1c_2 + c'\right)^2$, we have $W_m \subseteq C U'_1$. \qed

**Lemma 3.5.** If $N(\alpha_{m+1}) > \left(\frac{\delta N(\alpha_{m+1})^4}{2} + \sqrt{2c_1c_2^2} + \sqrt{c_1c_2}\right)^8$, then $C_mG_m \subseteq C_mU'_1$, i.e., condition II is satisfied.

**Proof.** By Lemma 3.3, in order to prove $C_mG_m \subseteq C_mU'_1$, it is sufficient to prove that for any $c \in C_m$ and $g \in G_m$, there exists a $\tilde{c} \in C_m$ such that $cg \equiv \tilde{c} \pmod{P_{m+1}}$ and $N(cg - \tilde{c}) < N^2(P_{m+1})$. So we should investigate when the inequality $N(cg - \tilde{c}) < N^2(P_{m+1})$ holds.

Let $c, \tilde{c} \in C_m, g \in G_m$, and let $M_1, M_2 \in \mathbb{R}$ with the conditions:

$$N(c) \leq M_1, \quad N(\tilde{c}) \leq M_1, \quad |g| \leq M_2, \quad |\sigma(g)| \leq M_2.$$

Then

$$N^\frac{1}{2}(cg - \tilde{c}) = |cg - \tilde{c}|\sigma(c)\sigma(g) - \sigma(\tilde{c})$$

$$\leq (|cg| + |\tilde{c}|)(|\sigma(c)|\sigma(g) + |\sigma(\tilde{c})|)$$

$$\leq |\sigma(c)|(|g\sigma(g)| + M_2(|c\sigma(\tilde{c})| + |\tilde{c}\sigma(c)|)) + |\tilde{c}\sigma(\tilde{c})|$$

$$= N^\frac{1}{2}(c)N^\frac{1}{2}(g) + M_2(|c\sigma(\tilde{c})| + |\tilde{c}\sigma(c)|) + N^\frac{1}{2}(\tilde{c}).$$

Let us estimate the term $|\sigma(c)| + |\tilde{c}\sigma(c)| = |\sigma(\tilde{c})| + \frac{N^\frac{1}{2}(c)N^\frac{1}{2}(\tilde{c})}{|\sigma(\tilde{c})|}$. By the definition of $C_m$, it is obvious that $|c| \leq c_1|\alpha_{m+1}|$ and $|c| = \frac{N^\frac{1}{2}(c)}{|\sigma(c)|} \geq \frac{1}{|c_2\sigma(\alpha_{m+1})|} N^\frac{1}{2}(c)$. So we have

$$\frac{N^\frac{1}{2}(c)}{c_2|\sigma(\alpha_{m+1})|} \leq |c| \leq c_1|\alpha_{m+1}|.$$

Similarly, we have

$$\frac{N^\frac{1}{2}(\tilde{c})}{c_1|\alpha_{m+1}|} \leq \frac{N^\frac{1}{2}(\tilde{c})}{|c|} \leq |\sigma(\tilde{c})| \leq c_2|\sigma(\alpha_{m+1})|.$$

Therefore

$$\frac{N^\frac{1}{2}(c)N^\frac{1}{2}(\tilde{c})}{c_1c_2N^\frac{1}{2}(\alpha_{m+1})} \leq |\sigma(c)| \leq c_1c_2N^\frac{1}{2}(\alpha_{m+1})$$
Let \( f(x) = x + \frac{N^\frac{1}{3}(c) N^\frac{1}{3}(\bar{c})}{x} \). It is easy to show that \( f(x) \) meets the maximal value at \( x = c_1c_2N^\frac{1}{3}(\alpha_{m+1}) \). So

\[
|c\sigma(c)| + |\bar{c}\sigma(\bar{c})| = |c\sigma(\bar{c})| + \frac{N^\frac{1}{3}(c) N^\frac{1}{3}(\bar{c})}{|c\sigma(\bar{c})|} \leq c_1c_2N^\frac{1}{3}(\alpha_{m+1}) + \frac{N^\frac{1}{3}(c) N^\frac{1}{3}(\bar{c})}{c_1c_2N^\frac{1}{3}(\alpha_{m+1})} \leq 2c_1c_2N^\frac{1}{3}(\alpha_{m+1}).
\]

Then

\[
N^\frac{1}{3}(cg - \bar{c}) \leq N^\frac{1}{3}(c) N^\frac{1}{3}(\bar{c}) + M_2(|c\sigma(c)| + |\bar{c}\sigma(\bar{c})|) + N^\frac{1}{3}(\bar{c}) \\
\leq M_1^\frac{1}{3}M_2^\frac{1}{3} + 2c_1c_2M_2N^\frac{1}{3}(P_{m+1}) + M_1^\frac{1}{3}
\]

By the definition of \( C_m \) and \( G_m \), we can take \( M_1 = c_1^2c_2^2N(\alpha_{m+1}) \) and \( M_2 = \delta N^\frac{1}{3}(\alpha_{m+1}) \). Hence we have

\[
N^\frac{1}{3}(cg - \bar{c}) \leq c_1c_2\delta^2N^\frac{1}{3}(\alpha_{m+1}) + 2\delta c_1c_2N^\frac{1}{3}(\alpha_{m+1}) + c_1c_2N^\frac{1}{3}(\alpha_{m+1}).
\]

So it is sufficient to consider the inequality

\[
c_1c_2\delta^2N^\frac{1}{3}(\alpha_{m+1}) + 2\delta c_1c_2N^\frac{1}{3}(\alpha_{m+1}) + c_1c_2N^\frac{1}{3}(\alpha_{m+1}) < N(\alpha_{m+1}),
\]

i.e.

\[
N^\frac{1}{3}(\alpha_{m+1}) - c_1c_2\delta^2N^\frac{1}{3}(\alpha_{m+1}) - 2\delta c_1c_2N^\frac{1}{3}(\alpha_{m+1}) - c_1c_2 > 0.
\]

This implies that when \( N(\alpha_{m+1}) > \left(\frac{\delta c_1c_2}{2} + \sqrt{\frac{\delta c_1c_2}{4} + \sqrt{c_1c_2}}\right)^4 \), we have \( N^\frac{1}{3}(cg - \bar{c}) < N(\alpha_{m+1}) \), as required. \( \square \)

### 3.3.2. Groenewegen’s general bound.

For any \( m \in \mathbb{Z}^+ \), we denote

\[
K_2U_m := (U_m \otimes U_m)/(a \otimes b|a, b \in U_m, a + b = 1 \text{ or } a + b = 0)
\]

and

\[
K_2^{(m)}O_F = \ker \left( K_2U_m \to \bigoplus_{Nv \leq Nv_m} k_v \right)
\]

It is clear that there is a natural map \( K_2U_m \to K_2F \). Moreover we write

\[
c_F = \max\{2^{2n}\rho d^2, 2^{2n/3}, \rho^{1/3}(\rho d^2)^{2/3}, \rho^{3}\}
\]

where

\[
d = \frac{2n\Gamma\left(\frac{n+2}{2}\right)}{(\pi n)^{n/2}} |\Delta|^{1/2}, \quad \rho = \left(\frac{2}{\pi}\right)^n |\Delta|^{1/2}
\]

and \( \rho \) is the packing density of an \( n \)-dimensional sphere. In [11], Groenewegen proved the following theorem.

**Theorem 3.6.** For every number field \( F \), for \( Nv_m > c_F \), the image of \( K_2^{(m)}O_F \) in \( K_2F \) is equal to the tame kernel of \( F \).

**Remark 3.7.** For an imaginary cyclic quartic field \( \mathbb{Q}\left(\sqrt{-(D + B\sqrt{D})}\right) \) of class number one (see Section 2), by Theorem 3.6 we can get a common bound of \( m \) for both condition I and condition II. But, from the computation of the next section, we know that for condition I the bound obtained by Lemma 3.4 is better than that obtained by Theorem 3.6 except for the cases \( B = 6, D = 37 \) and \( B = 2, D = \)
61, and for condition II, the bound obtained by Theorem 3.6 is better than that 
obtained by Lemma 3.5 except for the case $B = 1, D = 2$. The comparison of the 
results is listed in the following table.

**Table 1.**

| number field $F$ | Lemma 3.4 | Lemma 3.5 | Theorem 3.6 |
|------------------|-----------|-----------|-------------|
| $B = 1, D = 2$   | 172.525   | 3253.539  | 16146.993   |
| $B = 2, D = 13$ | 1173.677  | 45879.279 | 17321.1     |
| $B = 2, D = 29$ | 48710.067 | 1867701099.860 | 192289.567 |
| $B = 6, D = 37$ | 5284749.383 | 61546835.003 | 399362.147 |
| $B = 2, D = 53$ | 114166.647 | 4086894943.478 | 1173787.115 |
| $B = 2, D = 61$ | 180648285.891 | 1680328728.448 | 1789580.481 |

4. Decreasing the value $m$

4.1. The general idea. Let $F = \mathbb{Q}\left(\sqrt{-(D + B\sqrt{D})}\right)$ be an imaginary cyclic 
quartic field with class number $h_F = 1$ and $\xi$ the fundamental unit.

As Balabas and Gangl did, we also aim at decreasing theoretical bound of $m$
practically. The general idea is as follows.

At first, by Lemma 3.2, we get the constants $c_1, c_2, c'$. Let $c'' = \min\{(1 + cN + 
c')^2, cF\}$.

If $c'' < c_F$, there exists an $m_1 \in \mathbb{Z}^+$ satisfying $N(\mathcal{P}_{m_1}) < c''$ and $N(\mathcal{P}_{m_1+1}) > c''$. 
Thus, by Lemma 3.4, for $m \in \mathbb{Z}^+$ satisfying $m > m_1$ and $c'' < N(\mathcal{P}_{m_1+1}) \leq N(\mathcal{P}_m)$,
condition I holds for $m_1 + 1$. We want to show that it holds also for $m_1$.

If $c_F \leq c''$, there exists an $m'_1 \in \mathbb{Z}^+$ satisfying $N(\mathcal{P}_{m'_1}) \leq c''$ and $N(\mathcal{P}_{m'_1+1}) > c''$. 
By Theorem 3.6, the image of $K_2^{(m'_1+1)}\mathcal{O}$ in $K_2F$ is equal to the tame kernel of $F$.
However it is obvious that the image of $K_2^{(m'_1+1)}\mathcal{O}$ in $K_2F$ is $\ker(\partial : K_2F \rightarrow 
\prod_{v \in S_{m'_1+1}} \mathcal{O}_v^*)$. So it is necessary to show condition I holds for $m'_1$. Without loss of 
generality, we denote $m'_1$ also by $m_1$.

Similarly, from Lemma 3.5 or Theorem 3.6, there exists an $m_2 \in \mathbb{Z}^+$ such that condition II holds for $m_2 + 1$. We want to show that condition II holds also for $m_2$.

Then, for $m = m_1$ (resp. $m_2$), we will construct the subset $G_{m-1}, W_{m-1}$ and 
$C_{m-1}$ satisfying condition I (resp. condition II).

In this way, the value of $m$ can be decreased step by step.

4.2. Checking $\partial_m$ to be an isomorphism.

Our idea for checking $\partial_m$ to be an isomorphism is described as follows.

(I) Constructing the subset $W_{m-1}$.

Let $\xi$ be the fundamental unit. By (3.2), the subset

$$W_{m-1} = \{\alpha_1, \alpha_2, \cdots, \alpha_{m-1}\} \cup \{-1, \xi\}$$

needs to be defined, where $\alpha_i \in \mathcal{O}_F$ satisfies $\mathcal{P}_i = (\alpha_i)$ and $|\sigma(\xi)| \leq \left|\frac{\sigma(\alpha_i)}{\alpha_i}\right| \leq |\xi|$ 
for each $i = 1, 2, \cdots, m - 1$.

However, firstly for some fixed $i \in \{1, 2, \cdots, m - 1\}$, we must confirm that the 
generator $\alpha_i$ of the prime ideal $\mathcal{P}_i$ satisfies that $\left|\frac{\sigma(\alpha_i)}{\alpha_i}\right|$ nearly equals 1. Fortunately,
in the PARI library the function `GEN bnfisprincipal0(GEN bnf, GEN x, long flag)` can return such a generator \( \alpha_i \) for the prime ideal \( \mathcal{P}_i \). In fact, in the algorithm implemented by the above function, the generator has been reduced, which means that \( \frac{\sigma(\alpha_i)}{\alpha_i} \) nearly equals 1.

Secondly, we must get such \( \alpha_i \) for each \( i \leq m - 1 \). Thus, we must get at first the prime ideals whose norms are less than or equal to the boundary determined by Lemma 3.4 (Lemma 3.5 respectively). In fact, for each prime number \( p \in \mathbb{Z} \), it is easy to determine its residue class degree \( f_P \) and to obtain the prime ideals above it by the PARI function `GEN idealprimedec(GEN nf, GEN p, long f)`. So by iterating through the prime numbers which can be factored into the prime ideals with norm less than the boundary, we can get the required \( \alpha_i \) for each \( i = 1, 2, \cdots, m - 1 \).

(II) Constructing the subset \( G_{m-1} \).

For the only element \( g_{m-1} \in G_{m-1} \), we can know that

(i) \( g_{m-1}(mod \mathcal{P}_m) \) is the only generator of the multiplicative cyclic group \( \mathbb{k}_{v_m}^* \) of the residue class field \( \mathbb{k}_{v_m} \) by the second part of condition (ii) in Theorem 1.1;

(ii) the value \( \frac{g_{m-1}}{\sigma(g_{m-1})} \) should nearly equal 1, by the proof of Lemma 3.5.

In the case of \( f_{v_m} = 1 \), it is obvious that

\[
\langle g'_{m-1}(mod \mathcal{P}_m) \rangle = k_{v_m}^* \cong \mathbb{Z}/(\mathcal{P}_m \cup \mathbb{Z}) = \langle g'_{m-1}(mod \mathcal{P}_m \cup \mathbb{Z}) \rangle,
\]

where \( g'_{m-1} \in \mathbb{Z} \). Set \( g_{m-1} = g'_{m-1} \). Then we can get \( G_{m-1} = \{g_{m-1}\} \) with \( \frac{g_{m-1}}{\sigma(g_{m-1})} \approx 1 \).

In the case of \( f_{v_m} \neq 1 \), by the PARI function `GEN Idealstar(GEN nf, GEN ideal, long flag)`, the generator \( g_{m-1}(mod \mathcal{P}_m) \) of the cyclic group \( k_{v_m}^* \) can be obtained. So we can set \( G_{m-1} = \{g_{m-1}\} \). It is easy to show that the above condition (i) and (ii) are satisfied for the only element \( g_{m-1} \) of the set \( G_{m-1} \).

(III) Constructing the subset \( C_{m-1} \).

By (3.2), the subset \( C_{m-1} \) contains the lifting of all elements of the multiplicative group \( k_{v_m}^* \) and \( 1 \in F \). Moreover, by the proofs of Lemma 3.4 and Lemma 3.5, each element \( c_{m-1} \) of the set \( C_{m-1} \) should satisfy that the value \( \frac{c_{m-1}}{\sigma(c_{m-1})} \) nearly equals 1.

We can get the generator \( g_{m-1}(mod \mathcal{P}_m) \) of the group \( k_{v_m}^* \), so each element \( c_{m-1}(mod \mathcal{P}_m) \) of the group \( k_{v_m}^* \) can be expressed as

\[
c_{m-1,i}(mod \mathcal{P}_m) = (g_{m-1}(mod \mathcal{P}_m))^i
\]

where \( i = 1, 2, \cdots, N(v_m) - 1 \). But it is difficult to find a lifting \( c_{m-1,i} \) of the element \( c_{m-1}(mod \mathcal{P}_m) \), which satisfies that the value \( \frac{c_{m-1,i}}{\sigma(c_{m-1,i})} \) nearly equals 1.

The method we use to get a suitable lifting can be shown as follows.

Firstly, let \( c'_{m-1,i} = g_{m-1} - \beta \xi^k \) for each \( i = 1, 2, \cdots, N(v_m) - 1 \), where \( \beta \in \mathcal{O}_F \) and \( k \) is nonnegative integer.

Secondly, when \( \beta \) runs through the elements of \( \mathcal{O}_F \) in increasing order by norm and \( k \) runs through all nonnegative integers in increasing order, we can determine whether \( c'_{m-1,i} \in \mathcal{P}_m \) is true. Thus we can get the minimum \( \beta \) and \( k \) such that \( c'_{m-1,i} \in \mathcal{P}_m \) for each \( i = 1, 2, \cdots, N(v_m) - 1 \), and therefore \( \beta \xi^k \) is a lifting of \( c_{m-1,i}(mod \mathcal{P}_m) \). Hence, we can let \( c_{m-1,i} = \beta \xi^k \).

Lastly, we can obtain the set \( C_{m-1} = \{c_{m-1,i}|i = 1, 2, \cdots, N(v_m) - 1\} \cup \{1\} \).

(IV) Checking condition I(II).
After obtaining the subsets $W_{m-1}, G_{m-1}$ and $C_{m-1}$, now we can check condition I(II). Fortunately for us, the PARI function `GEN bnfissunit(GEN bnf, GEN sfu, GEN x)` can help us to check whether $\gamma \in U_m$ is true for some $\gamma \in \mathcal{O}_F$. Thus it is easy to write programme to check condition I(II) for the finite prime place $v_m$.

Using the above ideas, we can design the software architecture and algorithms and write a programme to compute some tame kernels $K_2\mathcal{O}_F$ for the cyclic quartic fields $F = \mathbb{Q}\left(\sqrt{-(D + B\sqrt{D})}\right)$ with class number one.

4.3. **Designing the classes.**

It is well known that Tate's theorem is right for any number field. Thus we can build a software architecture to be extensible and reusable for computing the tame kernel of a general number field, with the cases of imaginary cyclic quartic fields with class number one as examples. So in the following computation, firstly we will focus on all objects instead of the process.

All of objects are as follows:

1. The cyclic quartic field $F = \mathbb{Q}\left(\sqrt{-(D + B\sqrt{D})}\right)$;
2. The prime ideal $v_m$ of the algebraic integral ring $\mathcal{O}_F$;
3. The verification method which is used in this section;
4. The group of $S_m$-units $U_m = \{a \in F|v(a) = 0, v \notin S_m\}$;
5. Three subsets $C_{m-1}, W_{m-1}$ and $G_{m-1}$ of $U_{m-1}$ corresponding to $v_m$;
6. The constants $c_1, c_2$ corresponding to $F$.

Then, according to the objects and the relationships among them, we design the following three classes:

1. **CquarField** (an abstraction description of the field $F = \mathbb{Q}\left(\sqrt{-(D + B\sqrt{D})}\right)$);
2. **Cideal** (an abstraction description of the prime ideal $v_m$);
3. **Ccheck** (an abstraction description of the verification method).

Moreover, the constants $c_1, c_2$ are regarded as the attributes of **CquarField** and the sets $C_{m-1}, W_{m-1}, G_{m-1}$ as the the attributes of **Cideal**; an object of **CquarField** is regarded as an attribute of **Cideal**, which is actually an abstraction description about "prime ideal is subject to the cyclic quartic field $F$"; an object of **CquarField** is also regarded as an attribute of the class **Ccheck**, which means that "the verification method is corresponding to a given cyclic quartic field".

Summarily, the relations in the above descriptions can be indicated by the static class diagram given in Figure 2.

**Remark 4.1.** The reason why we use the Object-Oriented Programming (OOP) is that the architecture can be expanded. For example, if we can find a way to compute the tame kernel $K_2\mathcal{O}_{F_1}$ for another number field $F_1$, the only things we must do are:

1. Creating a class **CF$_1$** corresponding to $F_1$;
2. Creating a class **CF** as the parent class of **CF$_1$** and **CquarField**;
3. Making an object of **CF** as an attribute of **Cideal** and **Ccheck**.

Thus, we have complete the creation of the embryonic form of the architecture. The last work is to implement the classes.

4.4. **The methods of the three classes.** By the theory of the Object Oriented Programming, a class is partitioned into three parts: the name, the attributes and the methods.
For the above three classes, we have designed their methods, which are listed as follows (The algorithms implemented by these methods will be described in the next section):

(i) The methods of *CquarField*:

*return the constant c_1*/
GEN getc_1();

*return the constant c_2*/
GEN getc_2();

*return the transition matrix between the bases*/
GEN transMatrix();

*return the bound determined by Lemma 3.4*/
GEN getBoundOne();

*return the bound determined by Lemma 3.5*/
GEN getBoundTwo();

*return all prime ideals whose norms are less than*
*the bound which is determined by Lemma 3.4 (resp. Lemma 3.5)*
*and corresponds to the parameter num_condition 1 (resp. 2)*
GEN getPrimeTable(int num_condition);

(ii) The methods of *Cideal*:

*return the set W_{m-1} corresponding to the prime*
*ideal represented by the class Cideal*/
GEN getInitW();

*return the set G_{m-1} corresponding to the prime*
*ideal represented by the class Cideal*/
GEN getInitG();

*return all ideals whose norms are less than*
*the norm of the prime ideal represented by*
*the class Cideal*/
GEN getAllIdeal();

*return the set C_{m-1} corresponding to the prime*
*ideal represented by the class Cideal;*
*this is the parent thread function*/
GEN Para_getSetInitC();

*This is the child thread function*/
static void* Part_getSetInitC(void *arg);

*check condition 1 corresponding to the prime ideal*
*represented by the class Cideal*/
bool checkConditionOne();
ON THE TAME KERNELS OF IMAGINARY CYCLIC QUARTIC FIELDS WITH CLASS NUMBER ONE

*/check condition II corresponding to the prime ideal
*represented by the class Cideal*/
bool checkConditionTwo();

/*return the set \( U_{m} \) corresponding to the prime
*ideal represented by the class Cideal*/
GEN getUm();

(iii) The methods of Ccheck:
/*the parent thread function to check condition I */
bool Para_checkConditionOne(int num_thread);

/*the parent thread function to check condition II */
bool Para_checkConditionTwo(int num_thread);

/*the parent thread function to check condition I*/
static void* Part_checkConditionOne(void *arg);

/*the child thread function to check condition II*/
static void* Part_checkConditionTwo(void *arg).

4.5. Create the sequence diagram that shows the expected workflow. In order to show the process of computing the tame kernel of the number field \( F = \mathbb{Q}(\sqrt{A(D + B\sqrt{D})}) \), we create a sequence diagram given in Figure 3.

Remark 4.2. Some remark on the sequence diagram:

Firstly, we create an object of the class Ccheck, named as checker, by calling the constructed function Ccheck::check(int a,int b,int c,int d) of the class Ccheck, where the formal parameters \( a, b, c \) and \( d \) indicate the four parameters \( A, B, C \) and \( D \) of the cyclic quartic field \( F = \mathbb{Q}(\sqrt{A(D + B\sqrt{D})}) \), respectively. In the process, we create an object qfCom of the class CquarField, which indicates the cyclic quartic field \( F = \mathbb{Q}(\sqrt{A(D + B\sqrt{D})}) \). Moreover, some important invariants, such as the fundamental unit, the discriminant of the number field \( F \) and so on, of the cyclic quartic field \( F \) are obtained.

Secondly, after lots of tests we find some easy facts on the subset \( C_{s-1} \) of \( U_{s-1} \) corresponding to the prime place \( v_s \) of the number field \( F \) as follows.

1. In the process of obtaining the subsets \( C_{s-1}, G_{s-1} \) and \( W_{s-1} \) of \( U_{s-1} \), the most difficult one is to obtain \( C_{s-1} \):

2. The value of the theoretical bound, determined by the lemma 3.4, 3.5 and theorem 3.6, is very large. So the number of the sets \( C_{s-1} \) obtained by computing are also very large.

3. We suppose that some important information of tame kernel of the number field \( F \) must be hidden in the subset \( C_{s-1} \) of the set \( U_{s-1} \) for every prime ideal \( v_s \) of the number field \( F \).

Thus, it must take a long time to obtain the set \( C_{s-1} \) for every prime place \( v_s \) of the number field \( F \) whose norm \( N(v_s) \) is less than the theoretical bound. And we
think that it is a good idea to obtain the sets $C_{s-1}$ prior to the sets $G_{s-1}$ and $W_{s-1}$. Moreover, for finding more information on the tame kernel of the number field $F$ from those sets, we also hope that all of the obtained sets $C_{s-1}$ are preserved in persistent storage. Then, built on the above ideals, for every prime place $v_s$ of the number field $F$ whose norm $N(v_s)$ is less than the theoretical bound, after finishing creating the object checker, the sets $C_{s-1}$ are needed to get as follows.

(step 1) In order to obtain all of the set $C_{s-1}$, the method `bool Ccheck::Para_getSetC(int num_thread, int num_threadf)` is called, where the first(second) parameter means how many the threads are used for obtaining the set $C_{s-1}$ corresponding to the prime ideal $v_s$ with residue class degree $f_{v_s} = 1(f_{v_s} \neq 1)$.

(step 2) But the number of $v_s$, with norm $N(v_s)$ less than the theoretical bound, is very large. Then a technology of the parallel computing is needed. The method `void* Ccheck:: Part_getSetC(void *arg)` is child thread function.

(step 3) In the process of calling the method `Para_getSetC(int num_thread, int num_threadf)` to obtain all sets $C_{s-1}$, we must finish the following two works. One hand, it is necessary to get all prime numbers corresponding to the prime ideals of the number field $F$ whose norms are less than the theoretical bound, for which the method `GEN CquarField::getPrimeTable(int num_condition)` is designed; On the other hand, by the definition of $C'_s(3.2)$, we also must obtain all ideals of the number field $F$ whose norms are less than $c_1c_2N(v_{s+1})$. However, if using the PARI library function `GEN ideallist0(GEN nf, long bound, long flag)`, it will take a very long time to realizes the capability because the value $c_1c_2N(v_{s+1})$ is too large. For example, in the case of $F = \mathbb{Q}(\sqrt{-13+2\sqrt{13}})$, the theoretical bound is 45879 and we must take about 2.5 hours to obtain the ideals mentioned above; and in the case of $F = \mathbb{Q}(\sqrt{-29+2\sqrt{29}})$, the theoretical bound is 192289 and we must take about 60 hours. Moreover, there is no PARI function that returns all ideals whose norms are some $n \in \mathbb{Z}$ in pari library. So in order to minimize the consumption of time we must make use of the parallel computing in this procedure. Thus the methods `GEN CquarField::para_getAllideal(long num_thread,long num_condition)` and `void* CquarField::Part_getAllIdeal(void *arg)` must be designed as the father thread function and the child thread function respectively. The parameter `num_thread` means how many threads can be used for computing those ideals, and the parameter `num_condition` means on which condition the ideals are computed. By the two methods, it takes only about 10 minutes(resp. 1.5 hours) to obtain the ideals when $F = \mathbb{Q}(\sqrt{-13+2\sqrt{13}})$ (resp.$F = \mathbb{Q}(\sqrt{-29+2\sqrt{29}})$).

(step 4) In this step, by going through all prime ideals with the norms less than the theoretical bound, we obtain all of sets $C_{s-1}$. However, in every loop, we must finish the following works. Firstly, calling the constructed function `Cideal::Cideal(CquarField* quarf, GEN gen_prime, int i_th)` we can create the object of the class `Cideal` corresponding to prime ideal of the number field $F$ whose norm is less than the theoretical bound; secondly, calling the method `GEN Cideal::getSetInitG()` we obtain a generator element of the cyclic group $k^{*}_{v_s}$; lastly, calling the father thread function `void Cideal:: Para_getSetInitC()` and the child thread function `void* Cideal::Part_getSetInitC(void *arg)` we obtain the set $C_{s-1}$ and save as a text file.
Finally, to check condition I (resp. II), we use POSIX threads to design the father
thread function `bool Ccheck::Para_checkConditionOne(int num_thread)` (resp. `bool Ccheck::Para_checkConditionTwo(int num_thread)`) and the
child thread function `void* Ccheck::Part_checkConditionOne(void *arg)` (resp. `void* Ccheck::Part_checkConditionTwo(void *arg)`) to check condition I (resp. II).

### 4.6. The methods of the three classes.

By the theory of the Object Oriented Programming, a class is partitioned into three parts: the name, the attributes and
the methods.

For the above three classes, we have designed their methods, which are listed as follows (The algorithms implemented by these methods will be described in the
next section):

(i) The methods of `CquarField`:

- `GEN getc_1();`
- `GEN getc_2();`
- `GEN transMatrix();`
- `GEN getBoundOne();`
- `GEN getBoundTwo();`
- `GEN getPrimeTable(int num_condition);`

(ii) The methods of `Cideal`:

- `GEN getSetInitW();`
- `GEN getSetInitG();`
- `GEN fgetAllideal();`
- `GEN Para_getSetInitC();`
4.7. The algorithms implemented by the methods.

4.7.1. Some frequently-used algorithms. During decreasing the value $m$, there are three things we must compute from time to time. The first one is to decompose a (positive) prime number $p$ into prime ideals in the cyclic quartic field $F$, the second one is to obtain the generators of an ideal in the cyclic quartic field $F$ and the third one is to get the condition of determining whether an element of $\mathcal{O}_F$ is in the group $U_{m-1}$. However, the three things can be done by using Lemma 4.2, and Algorithm 4.1 and Algorithm 4.2 below. Moreover, as is well known, Lemma 4.1 can be implemented by the PARI’s functions \texttt{GEN \texttt{idealprimedec}(GEN \texttt{nf}, \texttt{GEN \texttt{p})}} and Algorithm 4.1 and Algorithm 4.2 by \texttt{GEN \texttt{bnfisprincipal0}(GEN \texttt{bnf, GEN bnf, GEN x, long flag)}} and \texttt{GEN \texttt{bnfissunit(GEN bnf, GEN sfu, GEN x)}}.

Lemma 4.3 (Theorem 4.8.13 ([8])). Let $F = \mathbb{Q}(\theta)$ be a number field, where $\theta$ is an algebraic integer, whose minimal polynomial is denoted $T(X)$. Let $f$ be the index of $\theta$. Then for any prime $p$ not dividing $f$ one can obtain the prime decomposition
of $p\mathcal{O}_F$ as follows. Let

$$T(X) \equiv \prod_{i}^{g} T_i(X)^{e_i} \pmod{p}\]$$

be the decomposition of $T$ into irreducible factors in $\mathbb{F}_p[X]$, where the $T_i(X)$ are taken to be monic. Then

$$p\mathcal{O}_F = \prod_{i=1}^{g} \mathcal{P}_i^{e_i},$$

where

$$\mathcal{P}_i = (p, T_i(\theta)) = p\mathcal{O}_F + T_i(\theta)\mathcal{O}_F.$$

Furthermore, the residual index $f_i$ is equal to the degree of $T_i(X)$.

---

**Algorithm 4.1** (Algorithm 6.5.10 ([8]))

**Require:** Given an ideal $I$ of $\mathcal{O}_F$ for a number field $F = \mathbb{Q}(\theta)$.

**Ensure:** Test whether $I$ is a principal ideal, and if it is, compute an $\alpha \in F$ such that $I = \alpha\mathcal{O}_F$.

1. **[Reduce to primitive]** If $I$ is not a primitive integral ideal, compute a rational number $a$ such that $I/(a)$ is primitive integral, and set $I \leftarrow I/(a)$.

2. **[Small norm]** If $N(I)$ is divisible only by prime numbers below the prime ideals in the factor base, set $v_i \leftarrow 0$ for $i < s$, $\beta \leftarrow a$ and go to step 4.

3. **[Generate random relations]** Choose random nonnegative integers $v_i < 20$ for $i < s$, compute the ideal $I_1 \leftarrow I \prod_{1 \leq i \leq s} S_i^{v_i}$, and let $J = I_1/(\gamma)$ be the ideal obtained by LLL-reducing $I_1$ along the direction of the zero vector. If $N(J)$ is divisible only by the prime numbers less than equal to $L_1$, set $I \leftarrow J$, $\beta \leftarrow a\gamma$ and go to step 4. Otherwise, go to step 3.

4. **[Factor I]** Using Algorithm 4.8.17 in [8], factor $I$ on the factor base $\mathcal{F}_B$. Let $I = \prod_{1 \leq i \leq k} F_i^{v_i}$. Let $X$ (resp. $Y$) be the column vector of the $x_i - v_i$ for $i \leq r$ (resp. $i > r$), where $r$ is the number of rows of the matrix $B$, as above, and where we set $v_i = 0$ for $i > s$.

5. **[Check if principal]** Let $Z \leftarrow D^{-1}U(X - BY)$ (since $D$ is a diagonal matrix, no matrix inverse must be computed here). If some entry of $Z$ is not integral, output a message saying that the ideal $I$ is not a principal ideal and terminate the algorithm.

6. **[Use Archimedean information]** Let $A$ be the $(c_1 + k)$-column vector whose first $c_1$ elements are zero, whose next $r$ elements are the elements of $Z$, and whose last $k - r$ elements are element of $Y$. Let $A_C = (a_i)_{1 \leq i \leq r_n} \leftarrow M_C^T A$.

7. **[Restore correct information]** Set $s \leftarrow \ln N(I)/n$, and let $A' = (a'_i)_{1 \leq i \leq n}$ be defined by $a'_i \leq \exp(s + a_i)$ if $i \leq r_1$, $a'_i \leftarrow \exp(s + (a_i - r_2))$ if $r_2 < i \leq n$.

8. **[Round]** Set $A'' \leftarrow \Omega^{-1}A'$ where $\Omega = \sigma_j(\omega_i)$ as in Algorithm 6.5.8 in [8]. The coefficients of $A''$ must be close to rational integers. If this is not the case, then either the precision used to make the computation was insufficient or the desired $\alpha$ is too large. Otherwise, round the coefficients of $A''$ to the nearest integer.

9. **[Terminate]** Let $\alpha'$ be the element of $\mathcal{O}_F$ whose coordinates in the integral basis are given by the vector $A''$. Set $\alpha \leftarrow \beta\alpha'$. If $I \neq \alpha\mathcal{O}_F$, output an error message stating that the accuracy is not sufficient to compute $\alpha$. Otherwise, output $\alpha$ and terminate the algorithm.
Algorithm 4.2 (Algorithm 7.4.8 ([9]))

Let $\text{Cl}(F) = (B, D_B)$ be the SNF of the class group of $F$, where $B = (b_i)$ and the $b_i$ are the ideals of $F$. The algorithm computes algebraic integers $\gamma_i$ for $1 \leq i \leq s$ such that $U_{S}(F) = U(F) \oplus_{1 \leq i \leq s} \mathbb{Z}\gamma_i$. We let $p_i$ be the prime ideals of $S$.

1. [Compute discrete logarithms] Using the principal ideal algorithm, compute the matrix $P$ whose columns are the discrete logarithms of $\overline{p}$ with respect to $B$, for each $p \in S$.

2. [Compute big HNF] Using one of the algorithms for HNF computations, compute the unimodular matrix $U = \left( \begin{array}{cc} U_1 & U_2 \\ U_3 & U_4 \end{array} \right)$ such that $(P|D_B)U = (0|H)$ with $H$ in HNF.

3. [Compute $\gamma O_F$] Compute the HNF $W$ of the matrix $U_1$, and set $[a_1, a_2, \ldots, a_s] \leftarrow [p_1, \ldots, p_s]W$.

4. [Find generators] Using the principal ideal algorithm again, for each $j$, find $\gamma_j$ such that $a_j = \gamma_j O_F$. Output the $\gamma_j$ and terminate the algorithm.

4.7.2. The algorithms implemented by the methods of CquarField. In the class CquarField, by Lemma 2.3, it is easy to design an algorithm implemented by the method GEN transMatrix() which returns the transition matrix from the basis $1, \beta, \beta^2, \beta^3$ to the basis $\gamma_0, \gamma_1, \gamma_2, \gamma_3$; similarly, by Lemma 3.2, it is also easy to design algorithms implemented by GEN getc_1(), GEN getc_2() which return the constants $c_1, c_2$ corresponding to the field $F = \mathbb{Q}\left(\sqrt{-D + B\sqrt{D}}\right)$; by Lemma 3.4 and Lemma 3.5, it is very easy to design an algorithm implemented by the methods GEN getBoundOne() and GEN getBoundTwo() which can be used to compute the bounds for condition I and condition II.

We must obtain all prime ideals whose norms are less than the bounds for condition I and condition II, which can be realized by the method GEN getPrimeTable(int num_condition) of CquarField.

In fact, let $b_1 = N(v_{m_0}), b_2 = N(v_{m_0'})$, and let

$$T_{F,i} = \{ p = \mathfrak{P} \cap \mathbb{Z} \in \mathbb{Z} | N(\mathfrak{P}) < b_i, \mathfrak{P} \in \text{spec}_F \},$$

$$T'_{F,i} = \{ \mathfrak{P} \in \text{spec}_F | N(\mathfrak{P}) < b_i \}, \quad i = 1, 2,$$

where $\text{spec}_F$ denotes the set of prime ideals of the cyclic quartic field $F$. To obtain the above sets, we design the following Algorithm 4.3 which can be implemented by the method GEN getPrimeTable(int num_condition) of CquarField.
Algorithm 4.3 (The algorithm on \texttt{getPrimeTable()})

1. \textbf{[Obtain the bound on norm]} By Lemma 4.5, Lemma 4.6 and Theorem 4.7, the bound on norm $b_1$ (resp. $b_2$) satisfying condition I (resp. condition II) can be obtained.

2. \textbf{[Obtain the set $T_{F,1}$ (resp. $T_{F,2}$)]} Using the PARI' function \texttt{GEN factoru(ulong n)}, the factorization of $n$ can be returned. Moreover, the result is a 2-component vector $[P, E]$, where $P$ and $E$ are the prime divisors of $n$ and the valuation of $n$ at prime point $p$ respectively.

3. \textbf{[Obtain the sets $T'_{F,1}$ (resp. $T'_{F,2}$)]} Using Algorithm 1, for any $k = p^s \in T'_{F,1}$ (resp. $T'_{F,2}$), the factorization ideals $P_i$ of $p$ can be obtained. By comparing the value $s$ with the norm of the ideal $P_i$, the elements of the set $T'_{F,1}$ (resp. $T'_{F,2}$) can be obtained.

4.7.3. The algorithms implemented by the methods of Cideal. In the class \texttt{Cideal}, the method \texttt{GEN getSetInitG()} can be used to compute the set $G_{m-1}$ corresponding to the object, prime ideal $v_m$, of \texttt{Cideal}. We can easily finish the codes of the method \texttt{GEN getSetInitG()} by using the PARI’s functions \texttt{GEN znstar(GEN n)} and \texttt{GEN idealstar0(GEN nf, GEN I, long flag)}, because the two functions have implemented the following well-known Algorithm 4.4.

Algorithm 4.4 (Algorithm 4.2.2 ([9]))

Let $m_0 = \prod P^{v_p}$ be an integral ideal, and assume that we are given the SNF of $(\mathcal{O}_F/\mathfrak{p}^{v_p})^\ast = (G, D)$. The algorithm computes the SNF of $(\mathcal{O}_F/m_0)^\ast$.

1. \textbf{[Compute $\alpha_p$ and $\beta_p$]} Using Extended Euclid Algorithm in Dedekind Domains (Algorithm 1.3.2 ([9])), compute $\alpha_p$ and $\beta_p$ such that $\alpha_p \in m_0/\mathfrak{p}^{v_p}, \beta_p \in \mathfrak{p}^{v_p}$ and $\alpha_p + \beta_p = 1$.

2. \textbf{[Terminate]} Let $G$ be the concatenation of the $\beta_p 1_{\mathcal{O}_F} + \alpha_p G_p$ and let $D$ be the diagonal concatenation of the SNF matrices $D_p$. Using the algorithm of SNF for Finite Groups (Algorithm 4.1.3 ([9])) on the system of generators and relations $(G, D)$, output the SNF of the group $(\mathcal{O}_F/m_0)^\ast$ and the auxiliary matrix $U_{\alpha}$, and terminate the algorithm.

In order to obtain the set $W_{m-1}$ corresponding to the prime ideal $v_m$, we design the method \texttt{GEN getSetInitW()} of \texttt{Cideal}. In the process of realizing this method, we use the PARI’s function \texttt{GEN idealist0(GEN nf, long bound, long flag)}, because they have implemented the following Algorithm 4.5 and returned all ideals whose norms are less than the value \texttt{bound}. We also give Algorithm 4.6, which outputs the set $W_{m-1}$ and is implemented by the method \texttt{GEN getSetInitW()}, as follows.
Algorithm 4.5 (Algorithm 2.3.23 ([9]))

Let $K$ be a number field and $B$ be a positive integer. The algorithm outputs a list $\mathcal{L}$ such that for each $n \leq B$, $\mathcal{L}_n$ is the list of all integral ideals of absolute norm equal to $n$.

1. [Initialize] For $2 \leq n \leq B$ set $\mathcal{L}_n \leftarrow \emptyset$, then set $\mathcal{L}_1 \leftarrow \mathcal{O}_K$ and $p \leftarrow 0$.

2. [Next prime] Replace $p$ by the smallest prime strictly larger than $p$. If $p > B$, output $\mathcal{L}$ and terminate the algorithm.

3. [Factor $p\mathcal{O}_K$] Using Algorithm 6.2.9 in [8], factor $p\mathcal{O}_K$ as $p\mathcal{O}_K = \prod_{1 \leq i \leq g}^{} \mathcal{P}_i^{e_i}$ with $e_i \geq 1$, and let $f_i = f(\mathcal{P}_i | p)$. Set $j \leftarrow 0$.

4. [Next prime ideal] Set $j \leftarrow j + 1$. If $j > g$, go to step 2. Otherwise, set $q \leftarrow p^{j^i}$, $n \leftarrow 0$.

5. [Loop through all multiples of $q$] Set $n \leftarrow n + q$. If $n > B$, go to step 4. Otherwise, set $\mathcal{L}_n \leftarrow \mathcal{L}_n \cup p_j \mathcal{L}_{n/q}$, where $\mathcal{L}_n$ is the list of products by the ideal $p_j$ of the elements of $\mathcal{L}_{n/q}$ and go to step 5.

Algorithm 4.6 (The algorithm on getSetInitW( ))

1. [Initialize] For the prime ideal $v_m$, set $W_{m-1} \leftarrow \emptyset$ and $I \leftarrow \emptyset$.

2. [Obtain ideals which norm are less than $N(v_m)$] Using the PARI’s function GEN ideallist0(GEN nf, long bound, long flag), all of ideals whose norms are less than $N(v_m)$ can be obtained. Then put them into the set $I$.

3. [Obtain all prime ideals whose norm are less than $N(v_m)$] By looping through the set $I$ and checking the structure of the ideal returned by PARI' function, we can get all prime ideals whose norm are less than $N(v_m)$.

4. [Obtain the set $W_{m-1}$] For the prime ideal $P_i$, by using the PARI’s function prime ideals whose norms are less than $N(v_m)$, the generator $\alpha_i$ can be returned. Then set $W_{m-1} \leftarrow \alpha_i$, where $i = 1, 2, \cdots, m$.

In the process of obtaining the sets $G_{m-1}$, $W_{m-1}$ and $C_{m-1}$, the most difficult thing is the computation of $C_{m-1}$, because we meet the following two difficulties:

(i) The set $C_{m-1}$ is too large when $N(v_m)$ is large since we have $|C_{m-1}| = N(v_m) - 1$;

(ii) we know that the set $C_{m-1}$ consists of the representatives of some elements in $k_{v_m}^*$, but we can not ensure that the set $C_{m-1}$ must satisfy condition I and condition II under arbitrary-chosen representatives.

To overcome these difficulties, we use the method of traversal but with the choice of representatives in a conjecturally right way.

Firstly, we find that the element $c \in C_{m-1}$ should be “some shortest distance point” in “some distance” of the set $c + v_m$. So in order to look for the right $c \in C_{m-1}$, we set the range from an element whose norm is one.

Secondly, it is our choice to take full advantage of multi-core processor hardware performance to reduce the computation time. Thus, we must use the technology of the multi-threaded parallel computing to improve the speed of Algorithm 4.7 obtaining the set $C_{m-1}$ as follows.
Algorithm 4.7 (The algorithm on \texttt{getSetInitC()})

Let \( v_m \) be a prime ideal of \( \mathcal{O}_F \). The algorithm outputs a set \( C_{m-1} \) satisfying the following conditions:

(i) The set \( C_{m-1} \) consists of the representatives of some elements in \( (k_{v_m})^* \);

(ii) For any element \( c \in C_{m-1} \), the equation \( N(c) = \min \{ N(c + t\alpha_m) | t \in \mathcal{O}_F \} \) holds.

1. \textbf{[Initialize]} Set \( \text{num\_good} \leftarrow 0 \) and \( C_{m-1} \leftarrow \emptyset \). Invoking the methods \texttt{GEN getc\_1()} and \texttt{GEN getc\_2()}, we can get the constant numbers \( c_1 \) and \( c_2 \), respectively. Moreover, invoking the \texttt{PARI}'s function \texttt{GEN ideallist0(GEN nf, long bound, long flag)} and \texttt{long pr\_get\_f(GEN pr)} , we can get the residue class degree \( f_m \) of \( v_m \) and the set \( C_m' \) of all ideals whose norms are less than or equal to \( (c_1 c_2)^2 N(v_m) \), respectively. Invoking the method \texttt{GEN Cideal::getSetInitG()} , we can get the unique element \( g \in G_{m-1} \). Lastly, set \( \text{num\_C'} \leftarrow |C_m'| \) and \( \text{num\_C} \leftarrow N(V_m) - 1 \).

2. \textbf{[Compare \text{num\_good} with \text{num\_C}]} If \( \text{num\_good}=\text{num\_C} \) holds, the algorithm return \( C_{m-1} \) and is terminated.

3. \textbf{[Set the germs of \( C_{m-1} \)]} For \( 1 \leq i \leq \text{num\_C} \), if \( f_m = 1 \) set \( \text{c\_i} \leftarrow i \). Otherwise, set \( \text{c\_i} \leftarrow g^i \).

4. \textbf{[Look for the appropriate elements in \( C_{m-1} \)]} For \( 1 \leq j \leq \text{num\_C'} \) and \( 1 \leq k \leq 70 \), set \( \text{c\_j} \leftarrow C_{m-1}[j] \). Then invoke the \texttt{PARI}'s function \texttt{long idealval(GEN nf, GEN x, GEN pr)} to decide whether or not \( c_i - c'_j \xi^k \) is in \( U_m \). Let the function’s returned value be \( b \). If \( b > 0 \), set \( c_i \leftarrow c'_j \xi^k \) and \( \text{num\_good} \leftarrow \text{num\_good} + 1 \), and go to step 2; otherwise, set \( j \leftarrow j + 1 \) and \( k \leftarrow k + 1 \).

In the class \texttt{Cideal}, the methods \texttt{GEN Para\_getSetInitC()} and \texttt{static void* Part\_getSetInitC(void *arg)} are the parent thread and the child thread respectively. Using these methods, we can obtain the set \( C_{m-1} \) corresponding to the prime ideal \( v_m \). Moreover we can change the number of the child threads with different computer’s hardware.

In the class \texttt{Cideal}, the two methods introduced above are \texttt{bool Cideal::checkConditionOne()} and \texttt{bool Cideal::checkConditionTwo()}. As a result, conditions I and condition II can be verified respectively for the prime ideal \( v_m \) by the two methods, which implement respectively Algorithm 4.8 and Algorithm 4.9 below.
Algorithm 4.8 (The algorithm on checkConditionOne())

Let $v_m$ be a prime ideal of $\mathcal{O}_F$. The algorithm check whether or not condition I is hold for $v_m$.

1. [Initialize] Set num_good ← 0. Invoking the methods GEN getSetInitW(), GEN getSetInitC() and getUm(), we can get the sets $W_{m-1}, C_{m-1}$ and $U_m$, respectively. Moreover, we can get the cardinal numbers of the sets $W_{m-1}, C_{m-1}$, denoted by num_W and num_C, respectively.

2. [Compare num_good with num_W] If num_good = num_W holds, the algorithm returns true and is terminated.

3. [Loop through all element in SetW] For $1 \leq i \leq num_W$, set $w_i ← W_{m-1}[i]$.

4. [Look for the appropriate element c in setC] For $1 \leq j \leq num_C$, set $c_j ← C_{m-1}[j]$. Invoking the PARI’s function GEN bnfissunit(GEN bnf, GEN sfu, GEN x) and getting it’s returned value $b$, we can decide whether or not $w_i c_j - 1$ is in $U_m$. More precisely, if $b > 0$, set num_good ← num_good + 1 and go to step 2; otherwise, set $j ← j + 1$.

Algorithm 4.9 (The algorithm on checkConditionTwo())

Let $v_m$ be a prime ideal of $\mathcal{O}_F$. The algorithm check whether or not condition II holds for $v_m$.

1. [Initialize] Set num_good ← 0. Invoking the methods GEN getSetInitG(), GEN getSetInitC() and getUm(), we can get the sets $G_{m-1} = \{g\}, C_{m-1}$ and $U_m$, respectively. Moreover, we can get the cardinal number of the set $C_{m-1}$, denoted by num_C.

2. [Compare num_good with num_C] If num_good = num_C holds, the algorithm returns true and is terminated.

3. [Loop through all element in SetC] For $1 \leq i \leq num_C$, set $c_i ← C_{m-1}[i]$.

4. [Look for the appropriate element $c'$ in setC] For $1 \leq j \leq num_C$, set $c'_j ← C_{m-1}[j]$. Invoking the PARI’s function GEN bnfissunit(GEN bnf, GEN sfu, GEN x) and getting it’s returned value $b$, we can decide whether or not $\frac{c_j}{g} - 1$ is in $U_m$, where $g$ is the unique element in set $G_{m-1}$. More precisely, if $b > 0$, set num_good ← num_good + 1 and go to step 2; otherwise, set $j ← j + 1$.

4.7.4. The algorithms implemented the methods of Ccheck. In the class Cideal, the method GEN getSetInitG() can be used to compute the set $G_{m-1}$ corresponding
to the object, prime ideal \( v_m \), of \( \text{Cideal} \). We can easily finish the codes of the method \text{GEN getSetInitG()} by using the \text{PARI}'s functions \text{GEN znstar(GEN n)} and \text{GEN idealstar0(GEN nf, GEN I, long flag)}, because the two functions have implemented the following well-known Algorithm 4.10.

**Algorithm 4.10 (Algorithm 4.2.2 ([9]))**

Let \( m_0 = \prod \mathfrak{p}_v^e \) be an integral ideal, and assume that we are given the SNF of \((\mathcal{O}_F/\mathfrak{p}_v^e)^* = (G_{\mathfrak{p}}, D_{\mathfrak{p}})\). The algorithm computes the SNF of \((\mathcal{O}_F/m_0)^*\).

1. **[Compute \( \alpha_{\mathfrak{p}} \) and \( \beta_{\mathfrak{p}} \)]** Using Extended Euclid Algorithm in Dedekind Domains (Algorithm 1.3.2 ([9])), compute \( \alpha_{\mathfrak{p}} \) and \( \beta_{\mathfrak{p}} \) such that \( \alpha_{\mathfrak{p}} \in m_0/\mathfrak{p}_v^e, \beta_{\mathfrak{p}} \in \mathfrak{p}_v^e \) and \( \alpha_{\mathfrak{p}} + \beta_{\mathfrak{p}} = 1 \).

2. **[Terminate]** Let \( G \) be the concatenation of the \( \beta_{\mathfrak{p}} 1_{\mathcal{O}_F} + \alpha_{\mathfrak{p}} G_{\mathfrak{p}} \) and let \( D \) be the diagonal concatenation of the SNF matrices \( D_{\mathfrak{p}} \). Using the algorithm of SNF for Finite Groups (Algorithm 4.1.3 ([9])) on the system of generators and relations \((G, D)\), output the SNF of the group \((\mathcal{O}_F/m_0)^*\) and the auxiliary matrix \( U_\alpha \), and terminate the algorithm.

In order to obtain the set \( W_{m-1} \) corresponding to the prime ideal \( v_m \), we design the method \text{GEN getSetInitW()} of \( \text{Cideal} \). In the process of realizing this method, we use the \text{PARI}'s function \text{GEN ideallist0(GEN nf, long bound, long flag)}, because they have implemented the following Algorithm 4.5 and returned all ideals whose norms are less than the value \text{bound}. We also give Algorithm 4.6, which outputs the set \( W_{m-1} \) and is implemented by the method \text{GEN getSetInitW()}, as follows.

**Algorithm 4.11 (Algorithm 2.3.23 ([9]))**

Let \( K \) be a number field and \( B \) be a positive integer. The algorithm outputs a list \( \mathcal{L} \) such that for each \( n \leq B \), \( \mathcal{L}_n \) is the list of all integral ideals of absolute norm equal to \( n \).

1. **[Initialize]** For \( 2 \leq n \leq B \) set \( \mathcal{L}_N \leftarrow \emptyset \), then set \( \mathcal{L}_1 \leftarrow \mathcal{O}_K \) and \( p \leftarrow 0 \).

2. **[Next prime]** Replace \( p \) by the smallest prime strictly larger than \( p \). If \( p > B \), output \( \mathcal{L} \) and terminate the algorithm.

3. **[Factor \( \mathcal{O}_F \)]** Using Algorithm 6.2.9 in [8], factor \( p\mathcal{O}_K \) as \( p\mathcal{O}_K = \prod_{1 \leq i \leq g} \mathfrak{p}_i^{e_i} \) with \( e_i \geq 1 \), and let \( f_i = f(\mathfrak{p}_i|p) \). Set \( j \leftarrow 0 \).

4. **[Next prime ideal]** Set \( j \leftarrow j + 1 \). If \( j > g \), go to step 2. Otherwise, set \( q \leftarrow p^j, n \leftarrow 0 \).

5. **[Loop through all multiples of \( q \)]** Set \( n \leftarrow n + q \). If \( n > B \), go to step 4. Otherwise, set \( \mathcal{L}_n \leftarrow \mathcal{L}_n \cup p_j \mathcal{L}_{n/q} \), where \( \mathcal{L}_n \) is the list of products by the ideal \( p_j \) of the elements of \( \mathcal{L}_{n/q} \) and go to step 5.
Algorithm 4.12 (The algorithm on \texttt{getSetInitW()})

1. \textbf{[Initialize]} For the prime ideal $v_m$, set $W_{m-1} \leftarrow \emptyset$ and $\mathcal{I} \leftarrow \emptyset$.
2. \textbf{[Obtain ideals which norm are less than $N(v_m)$]} Using the PARI's function \texttt{GEN ideallist0(GEN nf, long bound, long flag)}, all of ideals whose norms are less than $N(v_m)$ can be obtained. Then put them into the set $\mathcal{I}$.
3. \textbf{[Obtain all prime ideals whose norm are less than $N(v_m)$]} By looping through the set $\mathcal{I}$ and checking the structure of the ideal returned by PARI function, we can get all prime ideals whose norm are less than $N(v_m)$.
4. \textbf{[Obtain the set $W_{m-1}$]} For the prime ideal $\mathcal{P}_i$, by using the PARI's function prime ideals whose norms are less than $N(v_m)$, the generator $\alpha_i$ can be returned. Then set $W_{m-1} \leftarrow \alpha_i$, where $i = 1, 2, \ldots, m$.

In the process of obtaining the sets $G_{m-1}$, $W_{m-1}$ and $C_{m-1}$, the most difficult thing is the computation of $C_{m-1}$, because we meet the following two difficulties:

(i) The set $C_{m-1}$ is too large when $N(v_m)$ is large since we have $|C_{m-1}| = N(v_m) - 1$;

(ii) we know that the set $C_{m-1}$ consists of the representatives of some elements in $k^*_{v_m}$, but we can not ensure that the set $C_{m-1}$ must satisfy condition I and condition II under arbitrary-chosen representatives.

To overcome these difficulties, we use the method of traversal but with the choice of representatives in a conjecturally right way.

Firstly, we find that the element $c \in C_{m-1}$ should be “some shortest distance point” in “some distance” of the set $c+v_m$. So in order to look for the right $c \in C_{m-1}$, we set the range from an element whose norm is one.

Secondly, it is our choice to take full advantage of multi-core processor hardware performance to reduce the computation time. Thus, we must use the technology of the multi-threaded parallel computing to improve the speed of Algorithm 4.13 obtaining the set $C_{m-1}$ as follows.
Algorithm 4.13 (The algorithm on getSetInitC())

Let \( v_m \) be a prime ideal of \( \mathcal{O}_F \). The algorithm outputs a set \( C_{m-1} \) satisfying the following conditions:

(i) The set \( C_{m-1} \) consists of the representatives of some elements in \( (k_{v_m})^* \);
(ii) For any element \( c \in C_{m-1} \), the equation \( N(c) = \min \{ N(c + t\alpha_m) | t \in \mathcal{O}_F \} \) holds.

1. [Initialize] Set \( \text{num\_good} \leftarrow 0 \) and \( C_{m-1} \leftarrow \emptyset \). Invoking the methods \( \text{GEN getc\_1()} \) and \( \text{GEN getc\_2()} \), we can get the constant numbers \( c_1 \) and \( c_2 \), respectively. Moreover, invoking the \textsc{PARI}'s function \( \text{GEN ideallist0(GEN nf, long bound, long flag)} \) and \( \text{long pr\_get\_f(GEN pr)} \), we can get the residue class degree \( f_m \) of \( v_m \) and the set \( C'_m \) of all ideals whose norms are less than or equal to \( (c_1c_2)^2N(v_m) \), respectively. Invoking the method \( \text{GEN Cideal\_::getSetInitG()} \), we can get the unique element \( g \in G_{m-1} \). Lastly, set \( \text{num\_C'} \leftarrow |C'_m| \) and \( \text{num\_C} \leftarrow N(V_m) - 1 \).

2. [Compare num\_good with num\_C] If \( \text{num\_good} = \text{num\_C} \) holds, the algorithm return \( C_{m-1} \) and is terminated.

3. [Set the germs of \( C_{m-1} \)] For \( 1 \leq i \leq \text{num\_C} \), if \( f_m = 1 \) set \( c_i \leftarrow i \). Otherwise, set \( c_i \leftarrow g^i \).

4. [Look for the appropriate elements in \( C_{m-1} \)] For \( 1 \leq j \leq \text{num\_C'} \) and \( 1 \leq k \leq 70 \), set \( c'_j \leftarrow C'_{m-1}[j] \). Then invoke the \textsc{PARI}'s function \( \text{long idealval(GEN nf, GEN x, GEN pr)} \) to decide whether or not \( c_i - c'_j \xi^k \) is in \( U_m \). Let the function’s returned value be \( b \). If \( b > 0 \), set \( c_i \leftarrow c'_j \xi^k \) and \( \text{num\_good} \leftarrow \text{num\_good} + 1 \), and go to step 2; otherwise, set \( j \leftarrow j + 1 \) and \( k \leftarrow k + 1 \).

In the class \textit{Cideal}, the methods \( \text{GEN Para\_getSetInitC()} \) and \texttt{static void* Part\_getSetInitC(void *arg)} are the parent thread and the child thread respectively. Using these methods, we can obtain the set \( C_{m-1} \) corresponding to the prime ideal \( v_m \). Moreover we can change the number of the child threads with different computer’s hardware.

In the class \textit{Cideal}, the two methods introduced above are \texttt{bool Cideal::checkConditionOne()} and \texttt{bool Cideal::checkConditionTwo()}. As a result, conditions I and condition II can be verified respectively for the prime ideal \( v_m \) by the two methods, which implement respectively Algorithm 4.14 and Algorithm 4.15 below.
Algorithm 4.14 (The algorithm on checkConditionOne())
Let \( v_m \) be a prime ideal of \( \mathcal{O}_F \). The algorithm check whether or not condition I is hold for \( v_m \).

1. [Initialize] Set \( \text{num\_good} \leftarrow 0 \). Invoking the methods \( \text{GEN getSetInitW()} \), \( \text{GEN getSetInitC()} \) and \( \text{getUm()} \), we can get the sets \( W_{m-1}, C_{m-1} \) and \( U_m \), respectively. Moreover, we can get the cardinal numbers of the sets \( W_{m-1}, C_{m-1} \), denoted by \( \text{num\_W} \) and \( \text{num\_C} \), respectively.

2. [Compare \( \text{num\_good} \) with \( \text{num\_W} \)] If \( \text{num\_good} = \text{num\_W} \) holds, the algorithm returns true and is terminated.

3. [Loop through all element in SetW] For \( 1 \leq i \leq \text{num\_W} \), set \( w_i \leftarrow W_{m-1}[i] \).

4. [Look for the appropriate element \( c \) in setC] For \( 1 \leq j \leq \text{num\_C} \), set \( c_j \leftarrow C_{m-1}[j] \). Invoking the \textsc{PARI}'s function \( \text{GEN bnfissunit(GEN bnf, GEN sfu, GEN x)} \) and getting its returned value \( b \), we can decide whether or not \( w_i c_j - 1 \) is in \( U_m \). More precisely, if \( b > 0 \), set \( \text{num\_good} \leftarrow \text{num\_good} + 1 \) and go to step 2; otherwise, set \( j \leftarrow j + 1 \).

Algorithm 4.15 (The algorithm on checkConditionTwo())
Let \( v_m \) be a prime ideal of \( \mathcal{O}_F \). The algorithm check whether or not condition II holds for \( v_m \).

1. [Initialize] Set \( \text{num\_good} \leftarrow 0 \). Invoking the methods \( \text{GEN getSetInitG()} \), \( \text{GEN getSetInitC()} \) and \( \text{getUm()} \), we can get the sets \( G_{m-1} = \{g\}, C_{m-1} \) and \( U_m \), respectively. Moreover, we can get the cardinal number of the set \( C_{m-1} \), denoted by \( \text{num\_C} \).

2. [Compare \( \text{num\_good} \) with \( \text{num\_C} \)] If \( \text{num\_good} = \text{num\_C} \) holds, the algorithm returns true and is terminated.

3. [Loop through all element in SetC] For \( 1 \leq i \leq \text{num\_C} \), set \( c_i \leftarrow C_{m-1}[i] \).

4. [Look for the appropriate element \( c' \) in setC] For \( 1 \leq j \leq \text{num\_C} \), set \( c'_j \leftarrow C_{m-1}[j] \). Invoking the \textsc{PARI}'s function \( \text{GEN bnfissunit(GEN bnf, GEN sfu, GEN x)} \) and getting its returned value \( b \), we can decide whether or not \( c_i g c'_j - 1 \) is in \( U_m \) where \( g \) is the unique element in set \( G_{m-1} \). More precisely, if \( b > 0 \), set \( \text{num\_good} \leftarrow \text{num\_good} + 1 \) and go to step 2; otherwise, set \( j \leftarrow j + 1 \).

5. THE PROOF OF THEOREM 1.2

Let \( F = \mathbb{Q} \left( \sqrt{-(D + B \sqrt{D})} \right) \) be an imaginary cyclic quartic field. For the case \( B = 1, D = 2 \), invoking the method \( \text{GEN CquarField::getBoundOne()} \) (resp. \( \text{GEN CquarField::getBoundOne()} \)) we can know that for the prime ideals whose norms are greater than or equal to 172.525 (resp. 3253.529), condition I (resp. condition II) holds. Moreover, by invoking the method \( \text{bool Ccheck::Para\_checkConditionOne(int num\_thread)} \) (resp. \( \text{bool Ccheck::Para\_checkConditionOne(int num\_thread)} \)), it is proved that for the prime ideals whose norms are less than 172.525 (resp. 3253.529), condition I (resp. condition II) holds also.
Similarly, for the case $B = 2, D = 13$ we can show that

(i) the bound determined by Lemma 3.4 (resp. Theorem 3.6) is 1173.7 (resp. 17321.7);

(ii) for the prime ideals whose norms are less than 1173.7 (resp. 17321.7), condition I (resp. condition II) holds.

And for the case $B = 2, D = 29$ we can show that

(i) the bound determined by Lemma 3.4 (resp. Theorem 3.6) is 48710.1 (resp. 192289.6);

(ii) for the prime ideals whose norms are less than 48710.1 (resp. 192289.6), condition I (resp. condition II) holds.

For $F = \mathbb{Q}(\sqrt{-(13 + 2\sqrt{13})})$ or $\mathbb{Q}(\sqrt{-(29 + 2\sqrt{29})})$ by PARI/GP, we know that the torsion element is only $-1$. Hence, it is easy to show that $K_2\mathcal{O}_F$ can be generated by the two elements of order 2: $\{-1,-1\}, \{-1,\xi\}$, where $\xi$ is a fundamental unit of $F$.

However, in [3], Browkin proved the following formula:

$$2\text{-rank} K_2\mathcal{O}_F = r_1(F) + g(2) - 1 + 2\text{-rank} \left( \text{Cl}(F)/\text{Cl}_2(F) \right),$$

where $r_1(F)$ is the number of real places of $F$, $g(2)$ the number of primes over 2, and $\text{Cl}(F)$ the class group of $F$. It is well known that $\text{Cl}(F) = 1$ and that by PARI/GP, there is only one prime in $\mathcal{O}_F$ lying over 2. So the formula takes the form:

$$2\text{-rank} K_2\mathcal{O}_F = 0 + 1 - 1 + 0 = 0.$$ 

Hence there is no element of order 2. Thus the tame kernel $K_2\mathcal{O}_F$ is trivial. The proof is completed.

**Remark 5.1.** For $F = \mathbb{Q}(\sqrt{-(13 + 2\sqrt{13})})$ and $F = \mathbb{Q}(\sqrt{-(29 + 2\sqrt{29})})$, we keep a record of every $C_{m-1}$ in some text files, which can be found in [http://pan.baidu.com/s/1kVnSOCn](http://pan.baidu.com/s/1kVnSOCn) and [https://pan.baidu.com/s/1dFRn8ch](https://pan.baidu.com/s/1dFRn8ch) respectively.

**References**

1. H. Bass and J. Tate, *The Milnor ring of a global field*, Algebraic K-theory, II: įįClassicalįį algebraic K-theory and connections with arithmetic (Proc. Conf., Seattle, Wash., Battelle Memorial Inst., 1972), Springer, Berlin, 1973, pp. 349-446. Lecture Notes in Math., Vol. 342.
2. K. Belabas and H. Gangl, *Generators and relations for $K_2\mathcal{O}_F$*, K-Theory 31 (2004), no.3, 195-231.
3. J. Browkin, *The functor $K_2$ of the ring of integers of a number field*, in: Universal Algebra and Application, Banach Center Publ. 9, PWN, Warszawa, 1982,187-195.
4. J. Browkin and H. Gangl, *Tame and wild kernels of quadratic imaginary number fields*, Math. Comp. 68(1999) No.225, 291-306.
5. J. Browkin, *Computing the tame kernel of quadratic imaginary number fields*, Math. Comp., 69(2000) No.232, 1677-1683.
6. J. Browkin, *Construction of elements in Bloch group*, preprint.
7. J. Browkin, *Tame kernel of cubic cyclic fields*, Math. Comp. 74(2004) No.250, 967-999.
8. H. Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics, 138, Springer(1995).
9. H. Cohen, *Advanced topics in computational algebraic number theory*, Graduate Texts in Mathematics, 193, Springer(1995).
10. H. Garland, *A finiteness theorem for $K_2$ of a number field*, Ann. of Math. (2) 94 (1971), 534-548.
11. Richard. P. Georneffegen, *Bounds for computing the tame kernel*, Math.Comp. 73(2003) No.247, 1449-1458.
12. X. Guo, H. Qin and G. Song, *Computing the tame kernel of $\mathbb{Q}(\zeta_8)$*, Comm. Algebra 31(2003), no.2, 645-656.
13. K. Hardy, R. H. Hudson, D. Richman, K. S. Williams & N. M. Holtz, *Calculating the class numbers of imaginary cyclic quartic fields*, Carleton-Ottawa Mathematical Lecture Note Series 7(1986), pp 201.
14. R. H. Hudson, K.S. Williams *The integers of a cyclic quartic field*, Math.Comp. 20(1990) No.1, 145-150.
15. F. Keune, *On the structure of $K_2$ of the ring of integers in number fields*, K-theory 2(1989), 625-645.
16. S. Prata, *C++ prime plus*, Pearson Education Inc.
17. H. Qin, *Computation of $K_2\mathbb{Z}[\sqrt{-6}]$*, J.Pure Appl.Algebra. 96(1994),133-146.
18. H. Qin, *Computation of $K_2\mathbb{Z}[\frac{1\pm\sqrt{-11}}{2}]$*, Chin.Ann. of Math. 17B(1996), 63-72.
19. D. Quillen, *Higher K-Theory I*, In: Algebraic K-Theory 1. Lecture Notes in Mathematics, 341, 85-147, Springer (1973).
20. M. Skalba, *Generalization of Thue’s theorem and computation of the group $K_2\mathbb{Q}_F$*, J. Number Theory. 46(1994), no. 3, 303-322.
21. J. Tate, *Appendix*, Springer L.N.M. 342(1973), 429-446.
22. The PARI Group, PARI/GP version 2.7.5, Bordeaux, 2014, http://pari.math.u-bordeaux.fr/.
23. L. Zhang and K. Xu, *The tame kernel of $\mathbb{Q}(\zeta_5)$ is trivial*, Math. Comp. 85(2016) No.299, 1523-1538.

School of Mathematics and Statistics, Qingdao University, Qingdao 266071, P.R. China; Institute of Applied Mathematics of Shandong, Qingdao University, Qingdao 266071, P.R. China

E-mail address: zhanglong_note@hotmail.com

School of Mathematics and Statistics, Qingdao University, Qingdao 266071, P.R. China; Institute of Applied Mathematics of Shandong, Qingdao University, Qingdao 266071, P.R. China

E-mail address: kejianxu@amss.ac.cn