Fast Quantum Algorithms for Computing the Unit Group and Class Group of a Number Field

Sean Hallgren
NEC Laboratories America, Inc.
4 Independence Way
Princeton, NJ 08540
hallgren@nec-labs.com

March 3, 2005

Abstract

Computing the unit group and class group of a number field are two of the main tasks in computational algebraic number theory. Factoring integers reduces to solving Pell's equation, which is a special case of computing the unit group, but a reduction in the other direction is not known and appears more difficult. We give polynomial-time quantum algorithms for computing the unit group and class group when the number field has constant degree.

1 Introduction

Problems where quantum algorithms have found exponential speedups have mostly been of number theoretic origin. This line of research started with Simon's algorithm [11] and Shor's algorithms [10] for factoring and discrete log. More recently, quantum algorithms for Pell's equation and related problems were found [5]. The best known classical algorithm for Pell's equation is exponentially slower than that of factoring. Solving Pell's equation is a special case of the more general problem of finding the unit group of a number field.

A number field F can be defined as a subfield of the complex numbers $\mathbb{C}$ which is generated over the rational numbers $\mathbb{Q}$ by an algebraic number, i.e. $F = \mathbb{Q}(\theta)$ where θ is the root of a polynomial with rational coefficients. If F is a number field, then the subset of F consisting of all elements that are roots of monic polynomials (i.e. leading term 1) with integer coefficients, forms a ring $\mathcal{O}$, called the ring of integers of F. The ring $\mathcal{O} \subseteq F$ can be thought of as a generalization of $\mathbb{Z} \subset \mathbb{Q}$. In particular, we can ask whether $\mathcal{O}$ is a principal ideal domain, whether numbers in $\mathcal{O}$ have unique factorization, and what the set of invertible elements is. The unit group $\mathcal{O}^*$ is the set of invertible algebraic integers inside F, that is, elements $\alpha \in \mathcal{O}$ such that $\alpha^{-1} \in \mathcal{O}$.

Number fields are the central object of study in algebraic number theory. In his standard book on computational algebraic number theory, Cohen [3] states five main problems. In this paper we solve three of these problems for number fields of constant degree, generalizing the previous (real quadratic) case which worked for one dimension. In particular we compute the unit group, solve the principal ideal problem, and compute the class group, for constant degree number fields, in quantum polynomial-time.

While number fields are of great interest in number theory and in general mathematics, some of the most basic questions about them are not well understood yet. One example of an open problem is determining the class group of a number field, which is a finite abelian group attached to the number field. In fact, it is even open whether there exist infinitely many number fields with trivial class group. The question of the class group being trivial is also equivalent to asking whether the elements in the ring of integers $\mathcal{O}$ of the number field have unique factorization. It seems therefore that if quantum computers are built, then these algorithms provide an application for mathematicians to learn more about fundamental mathematics problems.

Number fields appear many places and have been studied for a long time. For example, Lenstra [6] points out that number fields appear in algorithms that do not even refer to number fields in the problem statement. The most notable example is probably the number field sieve, which is the best classical algorithm for factoring integers. Class groups were originally discovered by Gauss in 1798. Cohen [3] gives two applications of class groups. One is in factoring algorithms, where factoring reduces to computing the class group. Another is in potential proofs for Fermat's last theorem.

Classically, computing the unit group and the class group are closely related. The best algorithm for solving either one of these problems classically simultaneously solves the other. On the other hand, our quantum algorithm first computes the unit group, then the principal ideal problem algorithm uses the unit group algorithm, and then the class group algorithm uses the previous two algorithms.

The first issue that arises when computing the unit group has to do with extending the framework of the hidden subgroup problem. Most exponential speedups solve some hidden subgroup problem. This problem provides a nice way to classify known results and see how new problems relate to what is known. In the hidden subgroup problem, a group and a function on that group are given, with the promise that the function is constant and distinct on cosets of some subgroup. The goal is to compute the subgroup. Most of the success in solving this problem has been for abelian groups, including finite abelian groups, $\mathbb{Z}$, $\mathbb{Z}^n$, and $\mathbb{R}$. In this paper we extend the hidden subgroup problem to work for $\mathbb{R}^r$ for a constant r.

While there is a straightforward solution of the HSP over $\mathbb{Z}^n$, this does not appear to be the case for $\mathbb{R}^r$. The HSP over $\mathbb{Z}^n$ can be easily reduced to the HSP over $\mathbb{Z}$, which Shor solved, as follows. By restricting the function to an axis, an instance of the HSP over $\mathbb{Z}$ is created. Suppose that for each axis i, we solve the HSP instance and find that the period is p_i. Then the function restricted to $\mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n}$ is an HSP instance over a finite group, where the HSP algorithm works very cleanly. On the other hand, when working on the HSP in $\mathbb{R}^r$ where the vectors are real-valued, we will have to work in the full group $\mathbb{R}^r$.

One consequence of solving the HSP over $\mathbb{R}^r$ is a point of view that perhaps describes what quantum computers are able to solve in terms of familiar objects such as lattices and their algorithms. In particular, rather than using terminology about finding a hidden subgroup in $\mathbb{R}^r$, consider using the equivalent language of computing the basis of an unknown real-valued lattice, given access to a function which encodes the lattice. In the current setting, computing the unit group reduces to finding the basis of an unknown real-valued lattice. The algorithm will be stated in terms of this lattice, using the Fourier transform to sample from its dual lattice, and computing the dual of the dual of the lattice to arrive at a basis for the lattice. Proving this approach works involves showing how the Fourier transform behaves when it acts on only a finite piece of a lattice, and when the points only approximate the actual lattice vectors. We must show that the quantum state will still have enough information to reconstruct the lattice. The method we use only appears to work for a constant number of dimensions because the rounding introduces new noise into the distribution that is not present in the integer lattice case.

The second issue that arises when trying to compute the unit group is how to access information about the unit group and its associated lattice in polynomial-time. There is a large literature about doing computations in rings of integers in computational algebraic number theory. We were able to show that to compute the unit group in the HSP framework it is enough to compute certain reduced ideals of $\mathcal{O}$. We found an unpublished result in the thesis of Thiel [13] that allowed us to do this. This approach to accessing the lattice is also only polynomial-time for constant degree number fields because it computes the shortest vector of different lattices of dimension related to the degree of the number field during the computation.

Interestingly, these problems have been shown to be in NP ∩ Co-NP assuming the GRH, even for arbitrary degree number fields [12, 13]. Our algorithms here do not use any assumptions, except for when computing the class group where the GRH is needed to compute a set of generators for the group.

We leave as an open problem finding quantum algorithms to solve these problems for arbitrary degree number fields. That would likely involve handling the quantum limitation of too much noise in Fourier sampling because of the rounding of lattice vectors, as well as the classical limitation of computing reduced ideals.

The problems in this paper were also independently studied by Schmidt and Vollmer [8].

2 Background

The main object of study will be a number field F, i.e. a finite extension of $\mathbb{Q}$. For references about the computational aspects see [6, 3, 13]. As a field extension, F can be generated by a single element θ in $\mathbb{C}$, $F = \mathbb{Q}(\theta)$. The element θ is a root of a monic irreducible polynomial of degree n with rational coefficients, which is called the minimal polynomial of θ. The number n is called the degree of F (over $\mathbb{Q}$) and an extension of degree n is also an n dimensional $\mathbb{Q}$ vector space. As a basis for the vector space of $F/\mathbb{Q}$ we can choose $1, \theta, \theta^2, ..., \theta^{n-1}$.

There are different polynomial-time equivalent ways of describing the input, such as an approximation to θ, or by giving θ's minimal polynomial. The running time is in terms of the discriminant Δ of F, and is described below. There are five central computational problems associated with number fields as described in [3], and we will give polynomial-time quantum algorithms for three of them here.

The first problem is the unit group, where we want to compute a fundamental system of units for the finitely generated abelian group. The second is the class group, where we want to compute a the structure of a finite abelian group. The third problem is the principal ideal problem, which we will need to find the class group (and also breaks a cryptosystem [2] which can be based on constant degree number fields). Classically, the best known algorithms compute both the unit group and the class group at the same time.

First we define some properties associated with number fields. A number field $F = \mathbb{Q}(\theta)$ of degree n has n embeddings into $\mathbb{C}$, which we will denote by F_i. If $\theta_1, \dots, \theta_n$ are the roots of the minimal polynomial of θ, then the i-th embedding F_i is given by $\mathbb{Q}(\theta_i)$. In this way there are n ways to embed the number field in $\mathbb{C}$, and using θ was just one choice. Each real-valued root θ gives a real embedding, and each complex-valued θ gives a complex embedding. Let s denote the number of real embeddings and t the number of pairs of complex conjugate embeddings. Then $n = s + 2t$. Let $m = s + t$. An element in F has n conjugates, and F has m absolute values, all of which correspond to the embeddings. Given any number $\alpha \in F$, $\alpha = \sum_{i=0}^{n-1} a_i\theta^i$ for some rational numbers $a_i \in \mathbb{Q}$, let $\alpha^{(j)}$ denote the j-th conjugate of α, that is, the image of α in the j-th embedding: $\alpha^{(j)} = \sum_{i=0}^{n-1} a_i\theta_j^i$. The j-th absolute value $|\cdot|_j$ of a number α is a function of the absolute value in the j-th conjugate field: $|\alpha|_j = |\alpha^{(j)}|$ if the embedding is real, and $|\alpha|_j = |\alpha^{(j)}|^2$ if the embedding is complex.

Given the properties of number fields, we can move towards defining the unit group. The ring of integers is defined as the set of numbers in F that are the root of some monic polynomial in $\mathbb{Z}[x]$. Computing $\mathcal{O}$ is polynomial-time equivalent to finding the largest square factor of a given positive integer [6, Thm 4.4]. Since we are in the quantum setting we can compute $\mathcal{O}$ in polynomial time. The unit group $\mathcal{O}^*$ is the set of invertible elements in $\mathcal{O}$. In the trivial case of $F = \mathbb{Q}$ and $\mathcal{O} = \mathbb{Z}$, $\mathcal{O}^* = \pm 1$. By Dirichlet's unit theorem, the unit group in general will be isomorphic to $r = s + t - 1$ copies of $\mathbb{Z}$, together with an efficiently (in constant dimension) classically computable root of unity. Therefore computing the unit group $\mathcal{O}^*$ in our case will mean computing a fundamental system of units $\varepsilon_1, \dots, \varepsilon_r$ that generate $\mathcal{O}^*$. As discussed below this is not the entire story, yet, because these fundamental units may have exponentially many bits. In any case, they will satisfy that up to the root of unity μ, any $\varepsilon \in \mathcal{O}^*$ can be written as $\varepsilon = \mu\varepsilon_1^{k_1} \cdots \varepsilon_r^{k_r}$, where $k_1, \dots, k_r \in \mathbb{Z}$.

We now discuss how to represent F and $\mathcal{O}$, how to represent numbers in F, and which computations are polynomial-time with these numbers. The standard representation of a number field F with its ring of integers $\mathcal{O}$ is to specify a $\mathbb{Z}$ basis $\omega_1, \dots, \omega_n$ for $\mathcal{O}$, together with its multiplication table. That is, any number in $\alpha \in \mathcal{O}$ can be uniquely written as $\alpha = \sum_{i=1}^{n} a_i\omega_i$, where $a_1, \dots, a_n \in \mathbb{Z}$. This basis will also be a $\mathbb{Q}$ basis for F, so any $\alpha \in F$ can be uniquely written as $\alpha = \sum_{i=1}^{n} a_i\omega_i$, where $a_1, \dots, a_n \in \mathbb{Q}$. The multiplication table is a set of n^3 integers $(c_{ijk})_{ijk}$ that defines multiplication in F: $\omega_i\omega_j = \sum_{k=1}^{n} c_{ijk}\omega_k$. The discriminant Δ of F is the determinant of the matrix $(\mathrm{Tr}\ (\omega_i\omega_j))_{ij}$, where $\mathrm{Tr}\ : F \to \mathbb{Q}$ is the trace map. The multiplication table for $\mathcal{O}$ can be transformed so that it has $O(n^4(2 + \log|\Delta|))$ bits. For the purposes of analyzing running times it is customary to use Δ as the input, and an algorithm is polynomial time if it is polynomial in $\log|\Delta|$ and the degree n. In this representation, addition, multiplication, and division are polynomial time. However, representing a fundamental set of units for $\mathcal{O}^*$ is not polynomial size.

Compact representations allow a polynomial size representation of the fundamental units. Generally speaking, a compact representation of a number $\alpha \in F$ is a set of numbers $\gamma_1, \dots, \gamma_m \in F$ in the standard representation and $k_1, \dots, k_m \in \mathbb{Z}$, where the total amount of data is polynomial size, and $\alpha = \Pi_{i=1}^{m}\gamma_i^{k_i}$. This representation is not unique. Numbers in this compact representation can be added, multiplied, and divided provided that the number field is restricted to constant dimension. Another polynomial size representation is from the Log embedding into $\mathbb{R}^r$, defined by $\alpha \mapsto \text{Log } \alpha = (\log |\alpha|_1, \ldots, \log |\alpha|_r)$. These are irrational numbers, but keeping them to some precision will suffice. $L = \text{Log } \mathcal{O}^*$ is a lattice, and our main goal will be to find a basis for this lattice. Given a basis for this lattice, it is possible to compute compact representations for a fundamental system of units.

The final objects we will need are the (fractional) ideals of $\mathcal{O}$, denoted $\mathcal{I}$, together with minimal elements. An ideal I of $\mathcal{O}$ is a subset of $\mathcal{O}$ that is closed under addition and by multiplication by elements of $\mathcal{O}$. An ideal has a $\mathbb{Z}$ basis $\beta_1, \ldots, \beta_n$ and is represented by an integer matrix $(a_{ij})_{ij} \in \mathbb{Z}^{n \times n}$ where $\beta_i = \sum_{j=0}^{n} a_{ij}\omega_j$. A fractional ideal I is a set where dI is an ideal for some integer d, and representing fractional ideals includes the extra parameter d. The matrix $(a_{ij})_{ij}$ representing the ideal can be kept in HNF form, and in this way representation of ideals is unique. It is possible to multiply two ideals in polynomial time, and to multiply an ideal by an element of F. A minimal element $\mu \in I$ satisfies the condition that there does not exist a nonzero element $\alpha \in I$ such that $|\alpha|_i < |\mu|_i$ for all i. For a given ideal, there are an exponential number of minima in general. An ideal is reduced if 1 is a minimum, and this class of ideals is important because it is possible to keep the representation size bounded by a polynomial. We will typically use "ideal" for "reduced ideal" for ease, but it should be clear from the context which case it is. Given an ideal, finding the shortest lattice vector in the lattice defined by the image of $I \mapsto \{(\alpha^{(1)}, \ldots, \alpha^{(m)}) : \alpha \in I\}$ corresponds to finding a minimum of I. Thus to reduce a given ideal the shortest vector in an m dimensional lattice is computed, resulting in a minimum μ, and $\frac{1}{\mu}I$ will be a reduced ideal. Given a compact representation of $\alpha \in F$, αI is computable in polynomial time.

We will discuss the class group in section 5.

3 Computing the unit group

3.1 The function f hiding the unit group

The unit group of a number field is generated by r elements $\varepsilon_1, \ldots, \varepsilon_r$, which can have doubly exponential size. Under the Log map the unit group is the lattice $L = \text{Log } \mathcal{O}^* \subseteq \mathbb{R}^r$. Our goal is to compute a set of vectors which closely approximate a basis for L. This suffices to solve the problem in the Log representation, and since we are in the constant degree case, it is possible to compute a compact representation of the resulting units [13]. In this section we describe what information can be efficiently accessed about the lattice L. We will first describe a function that is an instance of the hidden subgroup problem over $\mathbb{R}^r$, i.e. there is some subgroup L (a lattice) such that $f(x) = f(y)$ if and only if $x - y \in L$. We next describe the discretized version which will be used in the algorithm, and show how to compute it in polynomial-time.

We wish to define a function f on $\mathbb{R}^r$ that hides L in a way useful for a quantum algorithm, i.e. such that $f(x) = f(y)$ if and only if $x - y \in L$. We do this by defining $f : \mathbb{R}^r \to \mathcal{I} \times \mathbb{R}^r$ by

$$f(x) = (I_x, \delta_x),$$

where $I_x = \frac{1}{\mu}\mathcal{O}$ is the reduced ideal such that μ is the minimum of $\mathcal{O}$ that minimizes $|\text{Log } \mu - x|$, and Log μ is less than x in each coordinate (if two are the same distance then sort them). The vector $\delta_x = x - \text{Log } \mu$ is the vector from x to Log μ of the reduced ideal $\frac{1}{\mu}\mathcal{O}$. The main problem is how to actually compute this function, or in particular, how to compute a specific ideal close to x.

The ideal is represented exactly by its integer matrix in HNF, and it's polynomial-size since it is reduced. The discretized function that is used in the quantum algorithm is defined as $f_N : \mathbb{Z}^r \to \mathcal{I} \times \mathbb{Z}^r$ defined by

$$f_N(i) = (I_{i/N}, k_{i/N})$$

$f(i/N) = (I_{i/N}, \delta_{i/N})$, and $(k_{i/N})_j = \lfloor N(\delta_{i/N})_j \rfloor$. This takes a cube cornered at $I_{i/N}$ which is large as possible while not containing any other reduced ideals, and slices it up into numbered cubes.

The main problem is how to compute I_x given x. Let the ideal $I = \alpha\mathcal{O}$ for some $\alpha \in F$, where $\text{Log } \alpha = x \in \mathbb{R}^r$. We will call x the distance of I (it has a distance from the origin). When multiplying two ideals their distances add, since $IJ = \alpha\mathcal{O} \cdot \beta\mathcal{O} = \alpha\beta\mathcal{O}$, and $\text{Log } \alpha\beta = \text{Log } \alpha + \text{Log } \beta$. The set of reduced ideals are more or less arbitrarily spaced, and they repeat modulo the lattice L. To compute an ideal near a point in space, ideals near the origin are computed, and repeated squaring is used. The issue that arises in this construction is that the product of two reduced ideals is not reduced. This ideal will not be too large, but continual multiplications can result in an ideal with exponentially large representation size. To fix this, a reduction step is performed after each ideal multiplication. Given an ideal I, the reduction step works by computing the shortest vector in the lattice $\underline{I} = \{(\alpha^{(1)}, \ldots, \alpha^{(m)})\}$. Any shortest vector in $\underline{I}$ corresponds to a minimal element μ of I, and $\frac{1}{\mu}I$ is a reduced ideal. The only guarantee we have is that the reduced ideal is within $\log|\Delta|$ of I. For the purposes of evaluation our function, we need to have the reduced ideal that is closest to I.

In one dimension (the real quadratic number field case), applying the reduction step to a reduced ideal results in the next reduced ideal in distance to the origin. Repeated application allows us to search for the desired ideal in polynomial time. We may therefore compute the exact ideal that we need for our function to have the required properties. In two or higher dimensions this process of applying reduction to reduced ideals does not work. We must find another way in order to compute the reduced ideal near the given point. There are two ways to solve this problem. The first is using an unpublished result of Theil [13]. In [13], Algorithm 6.2.20 will return all reduced ideals around a point in space. This method seems quite involved. A second more direct way to compute all reduced ideals around a point appears in the "Scan" algorithm in Schoof [9]. Both of these algorithms are polynomial-time for number fields with constant degree.

There is one technical issue that arises due to the rounding involved in the function. The above algorithms can be used to compute the distance of an ideal (given some crude approximation of its distance to begin with) to an arbitrary precision, however if the distance of the ideal falls very close to the point in question, it may not be possible to compute where it sits exactly. For example, in one dimension when computing the ideal to the left of $i/N \in \mathbb{R}$ it may not be possible to compute whether the ideal's distance is greater than or less than i/N if it is closer than the precision can handle (it is an open question whether some fixed precision works for all ideals). One way to fix this is to add a small random shift to the function, resulting in a function where no ideal lines up very close to a multiple of $1/N$. In one dimesion, if there are M reduced ideals in the range of the function being evaluated, then using

$$f'_N(i) = f_N(i + j/M^2) = (I_{i/N+j/(NM^2)}, k_{i/N+j/(NM^2)}),$$

where j is chosen randomly in $1..M^2$ satisfies this with high probability. Here f_N is extended to $\mathbb{Q}$ in the natural way. In higher dimesions, the same method works, by shifting the function a random amount in $\mathbb{R}^r$.

3.2 The algorithm

In this section we show how to compute a basis for a constant dimensional lattice hidden by a function, or in other words, how to solve some instances of the hidden subgroup problem over $\mathbb{R}^r$. We start with a definition of a general problem that is meant to include the function hiding the unit lattice defined in section 3.1 and the lattice defined in section 4 designed to encode the generator of a principal ideal.

Suppose there exists a function $f : \mathbb{R}^r \to S$ and an r dimensional lattice L, where r is a constant and S is a set, such that $f(x) = f(y)$ if and only if $x - y \in L$. A function $f_N : \mathbb{Z}^r \to S$ *hides* L if we may choose $N \in \mathbb{Z}$ such that a random point $i \in \mathbb{Z}_q^r$ satisfies the following condition with an inverse polynomial probability: for all $j \in \mathbb{Z}_q^r$, $f_N(i) = f_N(j)$ if and only if there is some vector $v \in L$ such that $|(i/N - j/N) - v| \leq 1/N$. Division by N is component-wise. In the case of the unit lattice, we choose $1/N$ to be 1/poly times the minimum distance between ideals. Notice that the value $(I, 0)$ will appear every time I appears in the range of the function, as will $(I, 1)$. However due to rounding if (I, i) has i near the border with the next reduced ideal, if the distance is close to i/N, then (I, i) may appear sometimes and not other times due to rounding. By our choice of N this bad border condition only happens a 1/poly fraction of the time.

Given a function hiding a lattice L we will show how to compute a basis for the dual lattice $L^{\perp}$. To compute a basis for L we need one more technical condition, which is that the lattice be well conditioned. A lattice is well conditioned if a matrix B whose columns form a basis for L is well conditioned, i.e. if $||B|| \cdot ||B^{-1}||$ is bounded.

We proceed to show how to find an ϵ-approximation to a basis of $L^{\perp}$. Let L_q be $L \cap [0, q)^r$. Let $[\cdot] : \mathbb{R} \to \mathbb{Z}$ be some function that maps each value $x \in \mathbb{R}$ either to $\lfloor x \rfloor$ or $\lceil x \rceil$ arbitrarily, and applied to a vector acts component-wise, each rounding up or down. While we will start in a way that looks familiar, with a superposition of points over a coset of a lattice (i.e. the lattice shifted by some vector), note that each point will only approximate the corresponding point of the coset of the lattice points in L_q:

$$\frac{1}{\sqrt{q^r}}\sum_{k\in\mathbb{Z}_q^r}|k, f_N(k)\rangle \rightarrow \frac{1}{\sqrt{|L_q|}}\sum_{\substack{v\in L, k_0+v\in[0,q)^r\\ k_0 \text{ fixed}}}|[N(k_0+v)]\rangle,$$

where the arrow represents measuring the function value which results in $f_N(k_0)$. We have $[Nk_0+Nv] = Nk_0+[Nv]$ since $Nk_0\in\mathbb{Z}$, and we are Fourier sampling, so we may ignore the coset representative Nk_0 during the analysis of the probabilities. Furthermore, states that are exponentially close behave the same way. We are left with

$$\frac{1}{\sqrt{|L_q|}}\sum_{\substack{v\in L_q, k_0+v\in[0,q)^r\\ k_0 \text{ fixed}}}|[Nv]\rangle.$$

Let M be the length of the longest basis vector in a basis for the lattice using bounds we have on a set of fundamental units. Since k_0 can be chosen in the parallelepiped defined by these basis vectors, the fraction of points within rM of the cube sides is exponentially small if q is chosen sufficiently large compared to M, and so the state is exponentially close to

$$\frac{1}{\sqrt{|L_q|}}\sum_{v\in L_q}|[Nv]\rangle.$$

Next Fourier sampling is done, i.e. compute the Fourier transform and measure. However, just Fourier sampling over $\mathbb{Z}_{qN}^r$ as usual does not appear to be enough to recover the dual lattice. To overcome this problem we "zero-fill", that is compute the Fourier transform over the larger domain $\mathbb{Z}_{qNk}^r$, with the additional part of the domain taking zero values. This type of operation has been studied asymptotically in [4] when the Fourier transform of the original distribution is understood. However, here the Fourier transform of the original distribution is not well understood (we need to zero-fill even to analyze it), and here we also need k to be a constant. Zero-filling was also used in Shor's algorithms, but only because he could not compute the desired transform at the time, not to get the analysis to work. After computing the Fourier transform over $\mathbb{Z}_{qNk}^r$ the state is:

$$\frac{1}{\sqrt{|L_q|}}\frac{1}{\sqrt{(qNk)^r}}\sum_{i\in\mathbb{Z}_{qNk}^r}\sum_{v\in L_q}\omega_{qNk}^{i\cdot[Nv]}|i\rangle.$$

Let $i=\lfloor kqw\rceil\in\mathbb{Z}^r$ and $w\in L^\perp\subseteq\mathbb{R}^r$. Here $\lfloor\cdot\rceil$ rounds to the closest integer and is applied to the vector component-wise. In the algorithm we will assume points are discarded if each coordinate does not satisfy $i_j\le\frac{qNk}{n}$ where $n=\log\Delta$, the problem size. In terms of w, we have that $|w_i|\le\frac{N}{n}+1$, so choosing larger values of N results in sampling more points of $L^\perp$. By choosing N large enough we ensure we have a large enough piece of the the dual to get a generating set. The inner product satisfies

$$\begin{aligned} i\cdot[Nv] &= (qkw+\delta_w)\cdot(Nv+\epsilon_v)\\ &= qNk(w\cdot v)+qk(w\cdot\epsilon_v)+\delta_w\cdot(Nv+\epsilon_v). \end{aligned}$$

The first term in the final sum is zero modulo qNk, since $v\cdot w\in\mathbb{Z}$ by definition of $L^\perp$. Considering the phase, the second term is

$$\frac{qk(w\cdot\epsilon_v)}{qNk}\le\frac{r\max|w_i|}{N}\le\frac{r}{n}.$$

The final term is also bounded:

$$\frac{\delta_w\cdot(Nv+\epsilon_v)}{qNk}\le\frac{\delta_w\cdot v}{qk}+\frac{\delta_w\cdot\epsilon_v}{qNk}\le\frac{r\max|v_i|}{qk}\le\frac{1}{8}$$

if $k\ge 8r$.

Since the maximum value in the phase is 1/4, the probability of measuring an integer vector $i \in \mathbb{Z}^r$ associated to a vector in the finite piece $L^{\perp}_{N/n}$ of the dual lattice is

$$\left| \frac{1}{\sqrt{|L_q|}} \frac{1}{\sqrt{(qNk)^r}} \sum_{v \in L_q} \omega_{qNk}^{i \cdot [Nv]} \right|^2$$

$$\geq \left| \frac{1}{\sqrt{|L_q|}} \frac{1}{\sqrt{(qNk)^r}} \frac{|L_q|}{2}(1+i) \right|^2 = \frac{|L_q|}{2(qNk)^r}.$$

Now we need to relate the number of points in L_q and $L^{\perp}_{N/n}$. Applying Proposition 8.7 in [7], we have that $|L_q| \geq \frac{q^r}{2 \det L}$ if $q \geq r^2 M$, and $|L^{\perp}_{N/n}| \geq \frac{(N/n)^r}{2 \det L^{\perp}}$ if $N \geq n^r r^2 M$. Therefore $|L_q||L^{\perp}_{N/n}| \geq \frac{q^r (N/n)^r}{4 \det L \det L^{\perp}} = \frac{q^r (N/n)^r}{4}$ and we have

$$\frac{|L_q|}{2(qNk)^r} \geq \frac{1}{8(nk)^r |L^{\perp}_{N/n}|}.$$

When such a point $i = \lfloor kqw \rceil$ is measured where $w \in L^{\perp}$, then $i/(qk) - w = (kqw + \delta_w)/(qk) - w = \delta_w/(qk)$, showing that $i/(qk)$ is $1/q$-close to w. Finally note that by definition $|L^{\perp}_{N/n}|$ is the number of points in the set we do not discard. We have shown:

Lemma 3.1. *Let M be the longest basis vector in some reduced basis of the lattice. Choose $N \geq n^r r^2 M$ and greater than $1/n$ times the minimum distance between ideals [13], and $q \geq r^2 M$. Suppose a function f_N hiding a lattice is Fourier sampled once over $\mathbb{Z}^r_{qNk}$, where points with any coordinate greater than qNk/n are discarded. With probability at least $\frac{1}{8(nk)^r}$ a point $i \in \mathbb{Z}^r_{qNk}$ results where $i/(kq)$ is $1/q$-close to a point in $L^{\perp}$.*

As can be seen from the lemma, choosing q larger increases the precision of the measured points, which will be important for matrix inversion in the algorithm below. Choosing N larger results less points in L_q, and hence more points in $L^{\perp}_{N}$, allowing enough points to be measured.

Algorithm 3.1 (Unit group algorithm).
Input: *Number field with unit group L*
Output: *A set of vectors approximating a basis for L*

```
1. Compute a basis for $L^{\perp}$
    (a) Fourier sample $f_N$ a constant number of times.
    (b) Compute a basis $B$ from the spanning set of vectors.
2. Compute a basis for $L$, by computing $B^{-T}$.
3. Verify that the results set are units [13], and repeat if not.
```

Theorem 1. *Algorithm 3.1 computes the unit group of a constant degree number field in quantum polynomial-time.*

Proof. Assume w.l.o.g. that points are as large as N (instead of N/n). By the proof of Lemma 3.1 the points are roughly uniformly sampled points from $L^{\perp}_{N}$.

We must show that the result is a generating set for $L^{\perp}$. First note that, using the notation from the last section, there must be a lattice point within M of a given corner of the cube $[0, N)^r$. Since this point may be outside the cube, pick a lattice point near each corner of the cube, with the i-th point within $2M$ of $(0, \ldots, 0, N - 2rM, 0, \ldots, 0)$ and inside the cube, where M is at coordinate i. Call the lattice generated by these points L'. Since L' is full dimensional, $L^{\perp}/L'$ is a finite group. With probability $|L^{\perp}/L'|/|L^{\perp}_{N}|$ a sample will be from this finite group according to Lemma 3.1. But $|L^{\perp}/L'| \geq \frac{(N-2rM)^r}{\det L^{\perp}}(1 - \frac{2rM}{N-2rM})$ and

$|L_N^\perp| \leq \frac{N^r}{\det L^\perp}(1+\frac{4rM}{N})$ so $|L^\perp/L'|/|L_N^\perp| \geq \frac{(N-2rM)^r}{N^r}\frac{N(N-4rM)}{(N+4rM)(N-2rM)} \geq \frac{1}{4}$ when $N \geq 4rM$, the sample is from a finite group for which we need a generating set.

Given a generating set, a basis can be computed using [1].

If the columns of B are a basis of $L^\perp$, then the columns of the inverse of B^T are a basis for L. The condition number of the matrix is large enough so that inversion is possible. The condition number may be bounded by using the cofactor formula for B, derived from B^{-T}. Since B^{-T} is a basis for L, B^{-T} is $1/\det L$ times the cofactor matrix. Since $\det L \geq 0.05$ [13], $1/\det L$ is bounded. As for the cofactor matrix, entry (i,j) is the determinant of a matrix formed by taking the matrix B^{-T} and delete row i and column j. Since B^{-T} has bounded entries, so does the matrix with deleted entries, and the determinant is at most the product of the length of the columns, which is therefore also bounded. □

4 The Principal Ideal Problem

In this section we will solve the principal ideal problem. One benefit of solving the more general HSP over $\mathbb{R}^r$ in the last section is that this section will be simpler than it was in [5]. Given an ideal I of $\mathcal{O}$ we wish to determine whether it is principal, and if it is to compute a generator, i.e. some $\alpha \in F$ such that $I = \alpha\mathcal{O}$. There is not a unique generator, since $\varepsilon I = I$ for any unit $\varepsilon \in \mathcal{O}^*$. Below we will describe an algorithm that computes α when I is principal. Given any ideal a candidate generator α can be computed by running the algorithm, and then computing $\alpha\mathcal{O}$ in polynomial time. The result is I if and only if I is principal.

Given a principal ideal $I = \alpha\mathcal{O}$ the goal is to compute α. As always a compact representation of α will be computed because the standard representation of α can have exponentially many bits. The first step is to compute a reduced ideal that is close by, together with the multiple, i.e. a reduced ideal J such that I is within $\log \Delta$ of J, say, together with $\beta \in F$ where $J = \frac{1}{\beta}I$.

Given a reduced principal ideal $I = \alpha\mathcal{O} = I_x$, where $x = \mathrm{Log}\ \alpha$ define the function $g : \mathbb{Z} \times \mathbb{R}^r \to \mathcal{I} \times \mathbb{R}$ by $g(a,y) = f(ax-y) = (I_{ax-y}, \delta_{ax-y})$, where $a \in \mathbb{Z}$ and $y \in \mathbb{R}^r$. The discretized version is $g_N : \mathbb{Z} \times \mathbb{Z}^r \to \mathcal{I} \times \mathbb{Z}$ defined by $g_N(a,b) = f(ax - b/N)$.

The ideal $I_{ax-b/N}$ can be computed in polynomial time by first computing I^a and $I_{-b/N}$ and then multiplying them. This function hides the lattice $\Lambda \subseteq \mathbb{Z} \times \mathbb{R}^r$, which is the set $\{(a,y) \subseteq \mathbb{Z} \times \mathbb{R}^r\ : ax - y \in L\}$, which has basis $(1,x)$ and r vectors $(0,v)$, where v runs over a basis for the unit lattice L.

When given a function that hides a constant dimensional lattice, the algorithm in the last section can find a basis for it in polynomial time. So assume that a basis for Λ has been computed, and the goal is to compute x. Since the first coordinate of vectors in Λ are integers, pick any two basis vectors of Λ that have first coordinates relatively prime, and compute the linear combination of them that has first coordinate equal to 1. At this point, we have a point $(1,y) \in \Lambda$, so $x - y \in L$, and therefore $y = \mathrm{Log}\ \varepsilon\alpha$ for some unit ε, where $I = \varepsilon\alpha\mathcal{O}$. To compute x, reduce y modulo the basis of L. This will give the particular set of coordinates for x. We can now conclude the following:

Theorem 2. *The principal ideal problem in a constant degree number field can be solved in quantum polynomial-time.*

5 Computing the Class Group

In this section we will show how to compute the class group of F. The class group Cl is a finite abelian group defined as the set of ideals modulo the set of principal ideals. By computing Cl, it is meant that the structure of the abelian group should be computed. The general setup will be the same as in [5], but we use the more general principal ideal problem algorithm that was solved in the last section, and we require a more complicated setup to compute a superposition over a set of ideals in an equivalence class since the ideals do not just sit on a line, but instead are in $\mathbb{R}^r$.

Decomposing abelian groups is one of the main tasks that can be done with quantum computation. Given a set of generators $\{g_1, \ldots, g_m\}$ for a group G, a polynomial-time multiplication algorithm, and a guarantee that each group element has a unique representation, decomposing an abelian group reduces to an instance of the abelian hidden subgroup problem as follows. Define $f : \mathbb{Z}^m \to G$ by $f(e_1, \ldots, e_m) = g_1^{e_1} \cdots g_m^{e_m}$. The hidden subgroup H is the lattice of relations: $H = \{(e_1, \ldots, e_m) : g_1^{e_1} \cdots g_m^{e_m} = \mathrm{id}\}$, where id is the group identity. It is easy to verify that f is an instance of the hidden subgroup problem, i.e. that $f(e) = f(d)$ if and only if $e - d \in H$. After computing a basis for the lattice H, the Smith Normal Form [7] gives the desired decomposition.

In the case of class groups of number fields the group elements are not known to have unique efficiently computable group representatives. However, we will show how to compute a set of orthogonal quantum states $|\phi_{I_1}\rangle, \ldots, |\phi_{I_k}\rangle$ where $|\phi_{I_i}\rangle$ is computed from two different ideals I and J if and only if they represent the same element of Cl. That is, $|\phi_I\rangle$ and $|\phi_J\rangle$ are orthogonal if I and J are different elements of Cl, and otherwise they are very large innerproduct. The idea is to accomplish this by defining $|\phi_I\rangle$ to be the superposition of all reduced ideals equivalent to I: $\sum_J |J\rangle$. More information will be necessary to actually create this. Then the usual hidden subgroup problem algorithm can be used, by replacing the function evaluation step $|e, f(e)\rangle$ by $|e, \phi_I\rangle$, where $I = g_1^{e_1} \cdots g_m^{e_m}$. Generators $g_1, \ldots, g_m$ for Cl can be chosen in polynomial-time assuming the GRH [13]. Multiplication of ideals (and so multiplication in Cl without unique representation) can be done in polynomial time in the same way that we have been doing, but since it is unknown how to compute a unique representative for the equivalence class of the group element, we will show how to compute a quantum state as described above.

To compute a superposition of reduced ideals equivalent to $I = g_1^{e_1} \cdots g_m^{e_m}$, start by computing the unit group of F, resulting in basis B. The columns of B span Log $\mathcal{O}^*$. Next the goal is to compute a superposition of reduced ideals whose distances are inside the parallopiped $\mathcal{P}$ defined by B. Using f_N from section 3.1 compute

$$\sum_{i \in \mathbb{Z}_N^r} |i, f_N(B \cdot i)\rangle = \sum |i, (I_{B \cdot i/N}, k_{B \cdot i/N})\rangle.$$

Next run the principal ideal problem algorithm with B on the ideal register and compute its offset i into a third register. Erase i in the left most register and then uncompute the principal ideal problem algorithm, erasing the new version of i in the third register, resulting in $\sum_{i \in \mathbb{Z}_N^r} |(I_{B \cdot i/N}, k_{B \cdot i/N})\rangle$, ignoring the zero in the first register. Using this as a subroutine in the hidden subgroup problem algorithm, we have shown:

Theorem 3. *The class group of a constant degree number field can be computed in quantum polynomial-time assuming the GRH.*

References

[1] J. Buchmann and M. Pohst. Computing a lattice basis from a system of generating vectors. In *Eurocal'87*, volume 378 of *LNCS*, pages 54–63. Springer-Verlag, June 1987.

[2] J. Buchmann and H. C. Williams. On the existence of a short proof for the value of the class number and regulator of a real quadratic field. In *Number theory and applications (Banff, AB, 1988)*, pages 327–345. Kluwer Acad. Publ., Dordrecht, 1989.

[3] H. Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993.

[4] L. Hales and S. Hallgren. Quantum fourier sampling simplified. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 330–338, Atlanta, Georgia, 1–4 May 1999.

[5] S. Hallgren. Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 653–658, Montreal, Quebec, Canada, 19–21 May 2002.

[6] H. W. Lenstra, Jr. Algorithms in algebraic number theory. *Bulliten of the American Mathematical Society*, 26(2):221–244, Apr. 1992.

[7] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science.* Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002.

[8] A. Schmidt and U. Vollmer. Polynomial time quantum algorithm for the computation of the unit group of a number field. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, May 2005. In these proceedings.

[9] R. Schoof. Computing Arakelov class groups. To appear, 2004.

[10] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, Oct. 1997.

[11] D. R. Simon. On the power of quantum computation. *SIAM J. Comput.*, 26(5):1474–1483, Oct. 1997.

[12] C. Thiel. Under the assumption of the generalized riemann hypothesis verifying the class number belongs to NP $\cap$ co-NP. In L. M. Adleman and M.-D. Huang, editors, *Algorithmic number theory, ANTS-I*, volume 877 of *Lecture Notes in Computer Science*, pages 234–247. Springer-Verlag, 1994.

[13] C. Thiel. *On the complexity of some problems in algorithmic algebraic number theory.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1995.