Asymptotically Optimal Topological Quantum Compiling

Vadym Kliuchnikov[†], Alex Bocharov[*], and Krysta M. Svore[*]
[†] *Institute for Quantum Computing and David R. Cheriton School of Computer Science, Univ. of Waterloo, Waterloo, Ontario (Canada)*
[*] *Quantum Architectures and Computation Group, Microsoft Research, Redmond, WA (USA)*

In a topological quantum computer, universality is achieved by braiding and quantum information is natively protected from small local errors. We address the problem of compiling single-qubit quantum operations into braid representations for non-abelian quasiparticles described by the Fibonacci anyon model. We develop a probabilistically polynomial algorithm that outputs a braid pattern to approximate a given single-qubit unitary to a desired precision. We also classify the single-qubit unitaries that can be implemented exactly by a Fibonacci anyon braid pattern and present an efficient algorithm to produce their braid patterns. Our techniques produce braid patterns that meet the uniform asymptotic lower bound on the compiled circuit depth and thus are depth-optimal asymptotically. Our compiled circuits are significantly shorter than those output by prior state-of-the-art methods, resulting in improvements in depth by factors ranging from 20 to 1000 for precisions ranging between 10^{-10} and 10^{-30}.

I. INTRODUCTION

As hardware devices for performing quantum computation mature, the need for efficient quantum compilation methods has become apparent. While conventional quantum devices will require vast amounts of error correction to combat decoherence, it is conjectured that certain quasiparticle excitations obeying non-abelian statistics, called non-abelian anyons, will require little to no error correction. Certain quantum systems based on non-abelian anyons intrinsically protect against errors by storing information globally rather than locally and can be used for computation [1]. If two quasiparticles are kept sufficiently far apart and their worldlines in $2+1$-dimensional space-time are braided adiabatically, a unitary evolution can be realized. One class of non-abelian anyons, called Fibonacci anyons, are predicted to exist in systems in a state corresponding to the fractional quantum Hall (FQH) plateau at filling fraction $\mu = 12/5$ [2, 3], where the theory is described by $SU(2)_k$ Chern-Simons-Witten theories [4–6] for $k = 3$. In fact, it has been shown that for $k = 3$ and $k > 4$, $SU(2)_k$ anyons will realize universal quantum computation with braiding alone [7, 8].

Previous work [9, 10] has developed methods, using the Solovay-Kitaev algorithm [11], for approximating a given single-qubit unitary to precision ε by a Fibonacci anyon braid pattern with depth $O(\log^c(1/\varepsilon))$, where $c \sim 3.97$ in time $t \sim \log^{2.71}(1/\varepsilon)$. For coarse precisions, one can also use brute-force search to find a braid with minimal depth $O(\log(1/\varepsilon))$ in exponential time [10]. Since the number of braids grows exponentially with the depth of the braid, this technique is infeasible for long braids, which are required to achieve fine-grain precisions. While the Solovay-Kitaev algorithm applies at any desired precision, including fine-grain precisions, it incurs a (costly) polynomial overhead that leads to compiled circuits of length $O(\log^{3+\delta}(1/\varepsilon))$, far from the asymptotic lower bound of $O(\log(1/\varepsilon))$.

In this work, we develop an algorithm for approximating a single-qubit unitary to precision ε by a Fibonacci anyon braid pattern with asymptotically optimal depth $O(\log(1/\varepsilon))$. Moreover, the algorithm requires only probabilistically polynomial runtime. We also classify the set of unitaries that can be represented exactly ($\varepsilon = 0$) by a Fibonacci braid pattern and present an algorithm for their compilation.

A high-level representation of our compilation algorithm is depicted in Figure 1. The algorithm takes as input an arbitrary single-qubit unitary operation and a desired precision ε and approximates the unitary with a special unitary gate that can be represented exactly by a Fibonacci anyon braid pattern. The exact synthesis algorithm is then applied to the special unitary gate and a $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit is synthesized. A Fibonacci anyon braid pattern can always be written as an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit and vice-versa, which we describe in detail in Section III. Finally, we rewrite the circuit as a braid pattern and apply peephole optimization [12].

As will be shown in Section III, in order for a unitary matrix to be exactly representable as a Fibonacci anyon braid pattern, its entries must come from a certain ring of algebraic integers. For that reason the approximation step is the most involved part of the algorithm. It consists of two stages: in the first stage, we find an algebraic number that is in ε-proximity to one of the elements of the input unitary matrix (Section V). This step can be viewed as a generalization of a numeric "round-off" operation. In the second stage, we complete the algebraic number with numbers from the same ring of algebraic integers such that all of the numbers form a unitary matrix. This is done by solving the relative norm equation (Section IV).

Solving the relative norm equation is in itself hard, and the task is further complicated by the fact that the equation is only solvable for a fraction of the generated "round-offs"; testing if the given "round-off" corresponds to a solvable norm equation is as hard as solving it. However, there is a fraction of "round-offs" that lead to easy instances of the problem and furthermore it is possible

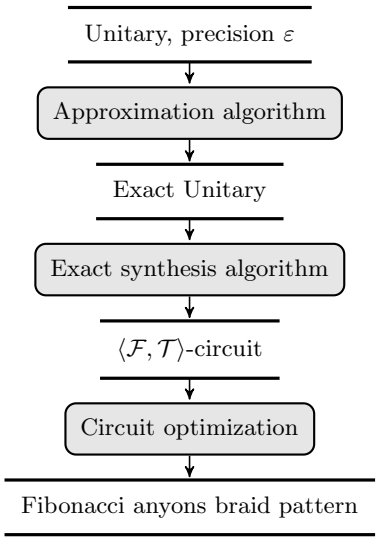

Figure 1: High-level flow of the compilation algorithm.

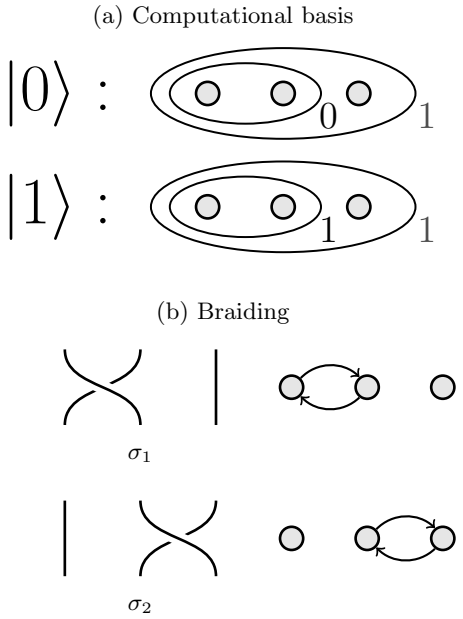

Figure 2: Encoding a qubit with three Fibonacci anyons. Grey circles corresponds to particles, numbers near ellipses show particles' collective charge [10].

to efficiently test if the given instances are easy.

Without sacrificing too much quality in the approximation, we aim to find these easy instances of the problem by randomly generating "round-offs" and testing if each instance is easy or not. We prove, under a given number theory conjecture (Conjecture 20), that the average number of iterations required to find an easy solvable instance is polynomial in the bit sizes of the inputs. We also provide numerical evidence that supports the conjecture. We carefully select the building blocks for the norm equation algorithm such that they also have probabilistically polynomial runtime for easy instances.

Our general approach to compilation into efficient single-qubit Fibonacci anyon circuits has been inspired in part by significant recent progress in efficient compilation into the $\langle H, T \rangle$ basis [13–15]) and the V basis [16]. The unifying theme behind these advances has been the use of algebraic number theory, in particular the theory of cyclotomic fields as the foundation for algorithms designed to meet the asymptotic lower bounds on the complexity of compiled circuits. Here, we employ a similar attack on circuit compilation through the use of algebraic number theory.

II. PRELIMINARIES

In this section, our main goal is to state the compilation problem and identify basic subproblems. We also provide relevant detail of the mathematical and computational properties of small anyon ensembles. We refer the reader to existing tutorials (such as [17, 18]) for detailed descriptions of the fundamental physics and mathematics of the anyon models.

There are several ways to simulate quantum circuits with Fibonacci anyon braid patterns. For each such simulation, we need to define how qubits are encoded with anyons. In this work, we focus on the three-particle encoding of a qubit [10], shown in Figure 2, where the computational basis state $|0\rangle$ corresponds to the first two anyons having collective charge zero, and state $|1\rangle$ corresponds to the first two anyons having collective charge one. The collective charge of all three anyons in both cases is one. Measurement of the collective charge of the first two anyons is a projective measurement in the computational basis. Unitary operations are realized by moving anyons around each other, where the result depends only on topological properties of anyon worldlines; the mathematical object that describes them is a braid group. Figure 2b shows anyon moves corresponding to the two generators σ_1, σ_2 of the three-strand braid group (corresponding to our the three particles). We also use σ_1, σ_2 to denote the corresponding unitary operations, where

$$\sigma_1 = \omega^6 \begin{pmatrix} 1 & 0 \\ 0 & \omega^7 \end{pmatrix}, \omega = e^{i\pi/5}.$$

It is convenient to express σ_2 with a Fusion matrix F, which corresponds to one of the parameters defining the Fibonacci anyon model:

$$F = \begin{pmatrix} \tau & \sqrt{\tau} \\ \sqrt{\tau} & -\tau \end{pmatrix}, \sigma_2 = F\sigma_1 F.$$

It has been shown [7, 8] that unitaries σ_1, σ_2 are approximately universal, e.g., for any single-qubit unitary and given precision ε there is a circuit consisting of σ_1 and σ_2 gates which approximates U within precision ε.

In this paper we describe an algorithm for approximating an arbitrary single-qubit unitary using circuits over a gate set consisting of σ_1, σ_2, σ_1^{-1}, and σ_2^{-1}. We call such a circuit a $\langle\sigma_1,\sigma_2\rangle$-circuit and refer to the four basis gates as σ gates. The circuit length (or equivalently depth) corresponds to the number of anyon moves needed to perform the given unitary. It defines how efficiently a given circuit implements a unitary.

More formally, the problem we solve can be stated as:

Problem 1. *Let $U \in U(2)$ be a single-qubit unitary and $\varepsilon > 0$ an arbitrary positive number. Find a $\langle\sigma_1,\sigma_2\rangle$-circuit c of length at most $O(\log(1/\varepsilon))$ such that the corresponding unitary is within distance ε from U.*

We use a global phase-invariant distance to measure the quality of approximation

$$d(U,V) = \sqrt{1 - |tr(UV^\dagger)|/2}.$$

Problem 1 can reduced to approximating two types of unitaries: rotations around the Z axis

$$R_z(\phi) := \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix}$$

and $R_z(\phi)X$, where X is the Pauli X gate. These two problems are in fact different because the X gate can only be approximated with a Fibonacci anyons braid pattern. The next lemma shows how the problem of approximating an arbitrary unitary can be reduced to these two types:

Lemma 2. *Consider a single-qubit unitary $U \in U(2)$. If the upper left entry of U is non-zero then it can be represented as*

$$e^{i\delta}\, R_z(\alpha)\, \mathcal{F}\, R_z(\beta)\, \mathcal{F}\, R_z(\gamma), \alpha,\beta,\gamma,\delta \in \mathbb{R}, \qquad (1)$$

otherwise it can be represented as $e^{i\delta}R_z(\phi)X$ for $\phi,\delta \in \mathbb{R}$.

Proof. First, we factor out an appropriate global phase from U to obtain a special unitary U'. The product $R_Z(\alpha)\,\mathcal{F}\,R_Z(\beta)\,\mathcal{F}\,R_Z(\gamma)$ corresponds to a special unitary with upper left entry

$$e^{-i\,(\alpha+\beta+\gamma)/2}\,\tau\,(e^{i\,\beta}+\tau)$$

which is always non-zero. It is not difficult to solve the equality $R_Z(\alpha)\,\mathcal{F}\,R_Z(\beta)\,\mathcal{F}\,R_Z(\gamma) = U'$ when U' has a non-zero upper left entry; in the other case it is not difficult to find ϕ such that $U' = R_z(\phi)X$. □

III. EXACT SYNTHESIS ALGORITHM

The goal of this section is two-fold: first, to classify all single-qubit unitaries that can be implemented exactly by braiding Fibonacci anyons and second, to describe an efficient algorithm for finding a circuit that corresponds to a given exactly implementable unitary. We start by introducing rings of integers and use them to describe the most general form of exactly implementable unitaries. Next, we introduce a complexity measure over the unitaries by using ring automorphisms. Finally, we describe the exact synthesis algorithm: a process, guided by the complexity measure, to find a circuit for an exactly synthesizable unitary.

For the purpose of this section it is more convenient to consider the elementary gates

$$\mathcal{T} = \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix}, \mathcal{F} = \begin{pmatrix} \tau & \sqrt{\tau} \\ \sqrt{\tau} & -\tau \end{pmatrix}$$

instead of σ_1 and σ_2. We call a circuit composed of $\mathcal{F},\mathcal{T}$ and $\omega\mathcal{I}$ gates an $\langle\mathcal{F},\mathcal{T}\rangle$-circuit, where $\mathcal{I}$ is the identity gate and $\omega\mathcal{I}$ is a global phase. The following relations imply that any unitary that is implementable by an $\langle\mathcal{F},\mathcal{T}\rangle$-circuit is also implementable by a $\langle\sigma_1,\sigma_2\rangle$-circuit and vice-versa:

$$\begin{array}{ll} \sigma_1 = (\omega\mathcal{I})^6\mathcal{T}^7, & \mathcal{T} = (\omega\mathcal{I})^2(\sigma_1)^3, \\ \sigma_2 = (\omega\mathcal{I})^6\mathcal{F}\mathcal{T}^7\mathcal{F}, & \mathcal{F} = (\omega\mathcal{I})^4\sigma_1\sigma_2\sigma_1. \end{array} \qquad (2)$$

For the applications considered in this paper the global phase $\omega\mathcal{I}$ is irrelevant.

Two rings are crucial for understanding our results. The first is the ring of integers extended by the primitive tenth root of unity ω

$$\mathbb{Z}[\omega] := \left\{ a + b\omega + c\omega^2 + d\omega^3 \;\middle|\; a,b,c,d \in \mathbb{Z} \right\}$$

and the second is its real subring

$$\mathbb{Z}[\tau] := \{ a + b\tau \mid a,b \in \mathbb{Z} \}.$$

It is not difficult to check that both definitions are correct: the sets defined above are both rings. It is straightforward to check that both sets $\mathbb{Z}[\omega]$ and $\mathbb{Z}[\tau]$ are closed under addition. To show that both sets are closed under multiplication it is sufficient to check the following equalities:

$$\omega^4 = -1 + \omega - \omega^2 + \omega^3,\ \omega^5 = -1,\ \tau^2 = 1 - \tau. \qquad (3)$$

Equality $\tau = \omega^2 - \omega^3$ implies that $\mathbb{Z}[\tau]$ is a subring of $\mathbb{Z}[\omega]$. Both rings are well studied in algebraic number theory.

In this section we use elementary methods that do not require any background. We discuss how objects and results of the section are related to the broader mathematical picture in Appendix A. For example, we show why $\mathbb{Z}[\tau]$ is a real subring of $\mathbb{Z}[\omega]$ (i.e., equal to $\mathbb{Z}[\omega]\cap\mathbb{R}$). For the purpose of this section it is sufficient to note that for any u from $\mathbb{Z}[\omega]$ its absolute value squared $|u|^2$ is from $\mathbb{Z}[\tau]$.

An *exact* unitary is defined by an integer k and u,v from $\mathbb{Z}[\omega]$ such that $|u|^2 + \tau|v|^2 = 1$ and

$$U[u,v,k] := \begin{pmatrix} u & v^*\sqrt{\tau}\,\omega^k \\ v\sqrt{\tau} & -u^*\,\omega^k \end{pmatrix}. \qquad (4)$$

The following Lemma explains the intuition behind the name.

n	u_n	$G(u_n)$
0	1	1
1	$\omega^2-\omega^3$	3
2	$2-\omega+\omega^3$	13
3	$-3+5\omega-2\omega^2-1$	57

Figure 3: Values of the Gauss complexity measure for the family of unitaries $U[u_n, v_n, k_n] = (\mathcal{FT})^n$, n from $\{0,1,2,3\}$.

Lemma 3 (Exact synthesis lemma)**.** *A unitary in $U(2)$ can be expressed as a product of $\mathcal{F}$ and $\mathcal{T}$ matrices if and only if it is exact.*

Proof. The "only if" part of the lemma is straightforward. Since $\tau \in \mathbb{Z}[\omega]$, both generators $\mathcal{T}$ and $\mathcal{F}$ have form (4) and multiplying a matrix of this form by either $\mathcal{T}$ or $\mathcal{F}$ preserves the form.

We defer the proof of the converse until we fully develop the exact synthesis algorithm (Figure 4) and prove its correctness. The "if" part of the lemma follows from that correctness proof (see Theorem 6). □

An automorphism of $\mathbb{Z}[\omega]$ that plays a fundamental role in the construction of the exact synthesis algorithm is the mapping

$$(.)^\bullet : \mathbb{Z}[\omega] \mapsto \mathbb{Z}[\omega] \text{ such that } \omega^\bullet = \omega^3. \tag{5}$$

By definition, a ring automorphism must preserve the sum and the product. For example, we find that

$$\tau^\bullet = (\omega^2-\omega^3)^\bullet = (\omega^2)^\bullet - (\omega^3)^\bullet = (\omega^\bullet)^2 - (\omega^\bullet)^3 = -\varphi.$$

Taking into account that $\varphi = \tau + 1$ (e.g., φ is from $\mathbb{Z}[\tau]$) we see that $(.)^\bullet$ restricted on $\mathbb{Z}[\tau]$ is also an automorphism of $\mathbb{Z}[\tau]$. The other important property of automorphism $(.)^\bullet$ is its relation to complex conjugation (which is another automorphism of $\mathbb{Z}[\omega]$):

$$(x^\bullet)^\bullet = x^*. \tag{6}$$

For example, this implies the equality $(x^*)^\bullet = (x^\bullet)^*$, which is used in several proofs in this work. It also implies that $|x^\bullet|^2 = (|x|^2)^\bullet$ because $|x|^2 = xx^*$.

The Gauss complexity measure G [19] is a key ingredient of the exact synthesis algorithm

$$G(u) := |u|^2 + |u^\bullet|^2. \tag{7}$$

We also extend the definition to exact unitaries as $G(U[u,v,k]) = G(u)$. Roughly speaking, the exact synthesis algorithm (Figure 4) reduces the complexity of the unitary U_r by multiplying it by $\mathcal{FT}^k$, while $G(U_r)$ describes how complex U_r is at each step. The example in Figure 3 illustrates the intuition behind G. The following proposition summarizes important properties of the Gauss complexity measure.

Input: U – exact unitary (i.e., in the form of (4))
procedure EXACT-SYNTHESIZE(U)
  $g \leftarrow G(U), U_r \leftarrow U, C \leftarrow$ (empty circuit)
  while $g > 2$ **do**
    $J \leftarrow \arg\min_{j\in\{1,\ldots,10\}} G\left(\mathcal{FT}^j U_r\right)$
    $U_r \leftarrow \mathcal{FT}^J U_r, g \leftarrow G(U_r)$
    Add $\mathcal{FT}^{10-J}$ to the beginning of circuit C
  end while
  Find k, j such that $U_r = \omega^k \mathcal{T}^j$
  Add $\omega^k \mathcal{T}^j$ to the beginning of circuit C
end procedure
Output: C – circuit over $\langle \mathcal{F}, \mathcal{T} \rangle$ that implements U

Figure 4: Exact synthesis algorithm.

Proposition 4. *For any x from $\mathbb{Z}[\omega]$, the Gauss complexity measure $G(x)$ is an integer, and there are three alternatives:*

(a) $G(x) = 0$, then $x = 0$,
(b) $G(x) = 2$, then $x = \omega^k$ for k – integer,
(c) $G(x) \geq 3$,

and $G\left(a+b\omega+c\omega^2+d\omega^3\right) \leq 5/2\left(a^2+b^2+c^2+d^2\right)$.

Proof. As observed in [19], $G\left(a+b\omega+c\omega^2+d\omega^3\right)$ is a positive definite quadratic form when viewed as a function of a,b,c,d. We found by direct computation that it takes integer values when a,b,c,d are integers, its minimal eigenvalue is $1/2$ and its maximal eigenvalue is $5/2$. This implies an upper bound on G in terms of a,b,c,d and an inequality

$$G\left(a+b\omega+c\omega^2+d\omega^3\right) \geq (a^2+b^2+c^2+d^2)/2.$$

Implication in (a) follows immediately from above. The inequality also allows us to prove the proposition by considering a small set of special cases

$$(a,b,c,d) \in S = \{-2,-1,0,1,2\}^4.$$

In all other cases, $G\left(a+b\omega+c\omega^2+d\omega^3\right)$ is greater than four, which corresponds to alternative (c). To finish the proof it is sufficient to exclude quadruples corresponding to $0, \omega^k$ from S and find that the minimum of $G\left(a+b\omega+c\omega^2+d\omega^3\right)$ over the remaining set is 3. □

To prove the correctness of the exact synthesis algorithm (Figure 4) we need the following technical result.

Lemma 5. *For any u, v from $\mathbb{Z}[\omega]$ such that $|u| \neq 1$, $|u| \neq 0$, $|u|^2 + \tau|v|^2 = 1$, there exists $k_0(u,v)$ such that:*

(a) $G\left((u+\omega^{k_0(u,v)}v)\tau\right)/G(u) < 1$,
(b) $G\left((u+\omega^{k_0(u,v)}v)\tau\right)/G(u) < \frac{\varphi^2}{2}(\sqrt[4]{5}-1)^2 + r(G(u))$, where $r(n)$ is in $O(1/n)$

In addition, for any k, the ratio $G\left((u+\omega^k v)\tau\right)/G(u)$ is upper bounded by $\frac{3\varphi^2}{2}(1+\sqrt{\tau})^2$.

We first prove that the exact synthesis algorithm is correct and efficient and then proceed to the proof of Lemma 5.

Theorem 6. *For any exact unitary U (in the form of (4)):*

1. *the exact synthesis algorithm (Figure 4) terminates and produces a circuit that implements U,*
2. *n, the minimal length of $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit implementing U, is in $\Theta(\log(G(U)))$,*
3. *the algorithm requires at most $O(n)$ arithmetic operations and outputs an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit with $O(n)$ gates*

Proof. The termination of the while loop in the algorithm follows from two facts: the complexity measure of U is an integer and, by Lemma 5, it strictly decreases at each step. Indeed, consider ratio $G\left(\mathcal{F}\mathcal{T}^j U_r\right)/G(U_r)$. If we denote the upper left entry of U_r by u and the lower left by $v\sqrt{\tau}$, then the ratio is precisely equal to the one considered in Lemma 5. By picking j that minimizes $G\left(\mathcal{F}\mathcal{T}^j U_r\right)$ we ensure that implications (a) and (b) of the Lemma hold. After the loop execution we have $G(U_r) \leq 2$. According to Proposition 4, the only possible values of G of the upper left entry u of U are either 0 or 2. There is no exactly synthesizable unitary with $u = 0$. In other words, there is no v from $\mathbb{Z}[\omega]$ such that equation $\tau|v|^2 = 1$ is solvable (see Section IV). The only remaining case is $G(u) = 2$. By Proposition 4, u must be a power of ω, therefore U_r must be representable as $\omega^k \mathcal{T}^j$. Correctness of the algorithm follows from the fact that during the algorithm execution at steps 3 and 6, it is always true that $U = U_C U_r$, where U_C denotes a unitary that is implemented by circuit C. This implies that any exactly synthesizable unitary U can be represented as a $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit.

Now we prove the second and the third statements. Taking into account that $\mathcal{F}^2 = \mathcal{I}$ and $\mathcal{T}^{10} = \mathcal{I}$, any exact unitary can be represented as the following matrix product

$$\omega^{k(m+1)} \mathcal{T}^{k(m)} \mathcal{F} \mathcal{T}^{k(m-1)} \mathcal{F} \mathcal{T}^{k(m-2)} \dots \mathcal{F} \mathcal{T}^{k(0)} \tag{8}$$

where $k(j)$ are from $\{0, \dots, 9\}$. The exact synthesis algorithm produces the circuit that leads precisely to representation (8) and m in this case is the number of steps performed by the algorithm. Lemma 5 implies that m is upper bounded by $\log_3(G(U)) + c_1$, as $\frac{\varphi^2}{2}(\sqrt[4]{5}-1)^2 < 1/3$. This gives an upper bound on the length of the circuit produced by the algorithm and also on n, the minimal possible length of any circuit that implements U.

Consider now representation (8) obtained from a minimal length $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit implementing U. We prove a bound on $G(U)$ by induction. First, by direct computation, $G\left(\mathcal{F}\mathcal{T}^{k(0)}\right)$ is equal to three. Next, we introduce $V_j = \mathcal{F}\mathcal{T}^{k(j-1)} \dots \mathcal{F}\mathcal{T}^{k(0)}$ and by the third part of Lemma 5 find

$$G\left(\mathcal{F}\mathcal{T}^{k(j)} V_j\right) \leq \frac{3\varphi^2}{2}(1+\sqrt{\tau})^2 G(V_j).$$

We note that multiplication by $\omega^{k(m+1)}\mathcal{T}^{k(m)}$ does not change the complexity measure and conclude that $\log(G(U))$ is upper bounded by some linear function of m. It is not difficult to see that m is less than n. This finishes the proof of the theorem. □

In Lemma 5 we show that the question about the change of the complexity measure is rather geometrical. On the high level, the relative change of complexity measure is related to an angle between certain vectors and the power of ω in the expression $G(\tau(u+\omega^k v))$ allows to adjust the angle by multiples of $\pi/5$.

Proof of Lemma 5. We first prove inequality (a). Consider the special case when there exists k such that $G\left((u+\omega^k v)\tau\right) = 2$. We define $k_0(u,v) = k$. Inequality (a) holds because $|u| \neq 1$ and $|u| \neq 0$ implies $G(u) \geq 3$ by Proposition 4.

Now we assume that $G\left((u+\omega^k v)\tau\right) \geq 3$. It is more convenient to consider ratio $\left|((u+\omega^k v)\tau)^\bullet\right|^2 / |u^\bullet|^2$ instead of $G\left((u+\omega^{k_0(u,v)}v)\tau\right)/G(u)$. The equality $|u^\bullet|^2 = G(u) - |u|^2$, inequalities $|u| < 1$, $G\left((u+\omega^k v)\tau\right) \geq 3$ and multiplicative property of $(.)^\bullet$ imply

$$\frac{G\left((u+\omega^k v)\tau\right)}{G(u)} < \frac{3\varphi^2}{2} \cdot \frac{\left|(u+\omega^k v)^\bullet\right|^2}{|u^\bullet|^2}.$$

Next we expand $\left|(u+\omega^k v)^\bullet\right|^2$ and see that it is equal to

$$|u^\bullet|^2 + |v^\bullet|^2 + 2\mathrm{Re}(u^\bullet (v^\bullet)^* \omega^{-3k}).$$

Here we used that $\omega^\bullet = \omega^3$. It is convenient to introduce

$$e^{i\phi} := \frac{u^\bullet}{|u^\bullet|}\left(\frac{v^\bullet}{|v^\bullet|}\right)^* \quad \alpha := \frac{|v^\bullet|^2}{|u^\bullet|^2}.$$

In terms of $\alpha, e^{i\phi}$ we find

$$\frac{\left|(u+\omega^k v)^\bullet\right|^2}{|u^\bullet|^2} = 1 + \alpha + 2\mathrm{Re}\left(e^{i(\phi - 3\pi k/5)}\right)\sqrt{\alpha}. \tag{9}$$

It is always possible to chose $k_0(u,v)$ such that

$$\frac{\left|(u+\omega^{k_0(u,v)} v)^\bullet\right|^2}{|u^\bullet|^2} < 1 + \alpha - 2\sqrt{\alpha}\cos(\pi/10). \tag{10}$$

Using the estimate above we get

$$\frac{G\left((u+\omega^{k_0(u,v)}v)\tau\right)}{G(u)} < \frac{3\varphi^2}{2} \cdot \left(1 + \alpha - 2\sqrt{\alpha}\cos(\pi/10)\right). \tag{11}$$

Note that α is a function of $|u^\bullet|^2$. Indeed, $|u|^2+\tau\,|v|^2=1$ implies

$$1=(|u|^2+\tau\,|v|^2)^\bullet=|u^\bullet|^2-\varphi\,|v^\bullet|^2\,.$$

We conclude that $\alpha=\tau(|u^\bullet|^2-1)/\,|u^\bullet|^2$. Ratio $G\left((u+\omega^{k_0(u,v)}v)\tau\right)/G\,(u)$ is upper bounded by the value of the right hand side of inequality (11) at $|u^\bullet|^2=2$ which is approximately 0.9982. Indeed, the right hand side is a monotonically decreasing function of $|u^\bullet|^2$. In addition, it is always the case that $|u^\bullet|^2>2$. This follows from the relation between $|u^\bullet|^2$ and $G\,(u)$ and the inequality $G\,(u)\geq 3$. This completes the proof of (a).

To show (b) we first note that

$$\frac{G\left((u+\omega^k v)\tau\right)}{G\,(u)}<\frac{G\left((u+\omega^k v)\tau\right)}{|((u+\omega^k v)\tau)^\bullet|^2}\cdot\frac{\varphi^2\left|(u+\omega^k v)^\bullet\right|^2}{|u^\bullet|^2}.$$

To complete the proof of (b) it is sufficient show that the first ratio in the right hand side of the inequality above is always less than $1+r_1(G\,(u))$ and the second one is less than $\frac{\varphi^2}{2}(\sqrt[4]{5}-1)^2+r_2(G\,(u))$ when $k=k_0(u,v)$, for $r_{1,2}(n)$ from $O(1/n)$. The definition of $G\,(.\,)$ implies

$$\frac{G\left((u+\omega^k v)\tau\right)}{|(u+\omega^k v)^\bullet|^2}<1+1/\left|(u+\omega^k v)^\bullet\right|^2.$$

It follows from equation (9) that there exists C_1 such that $1/\left|(u+\omega^k v)^\bullet\right|^2<C_1/\,|u^\bullet|^2$. We conclude that $r_1(x)=C_1/(x-1)$. Next we rewrite the right hand side of equation (10):

$$\frac{\left|(u+\omega^{k_0(u,v)}v)^\bullet\right|^2}{|u^\bullet|^2}<1+\tau+2\sqrt{\tau}\cos(\pi/10)+f(|u^\bullet|^2).$$

Taking into account that f is monotonically decreasing and $|u^\bullet|^2>G\,(u)-1$, we define $r_2(x)=f(x-1)$. By direct computation, we note that

$$1+\tau+2\sqrt{\tau}\cos(\pi/10)=\frac{1}{2}(\sqrt[4]{5}-1)^2$$

which completes the proof of (b).

Now we obtain the bound for arbitrary k. We use equation (9) and inequality $\alpha<\tau$ to find

$$\left|(u+\omega^k v)^\bullet\right|^2/|u^\bullet|^2\leq(1+\sqrt{\tau})^2.$$

As before, by distinguishing cases when $G\left((u+\omega^k v)\tau\right)$ is equal to two or greater than two we get

$$\frac{G\left((u+\omega^k v)\tau\right)}{G\,(u)}<\frac{3\varphi^2}{2}(1+\sqrt{\tau})^2$$

which finishes the proof. □

IV. NORM EQUATION

In this section, we begin by explaining the role of norm equations in the approximate synthesis of Fibonacci anyon circuits and proceed with a detailed description of solvability conditions and the essential procedures required for solving norm equations. In the concluding subsection, we present an algorithm for solving a certain class of norm equations in probabilistically polynomial runtime.

A. Motivation

As outlined in Section I, a method for completing an element from the ring $\mathbb{Z}\,[\omega]$ by other elements from the ring is required, such that together they make up a unitary matrix of the form (4).

For example, consider first the special case where our compilation target is the Pauli X gate

$$X=\begin{pmatrix}0&1\\1&0\end{pmatrix}.$$

As per (4), the off-diagonal element 1 needs to be represented as $1=v\sqrt{\tau}, v\in\mathbb{C}$, which makes v equal to $\sqrt{\tau^{-1}}=\sqrt{\phi}$. Since the latter does not belong to $\mathbb{Z}\,[\omega]$ it must be approximated by an element of $\mathbb{Z}\,[\omega]$. Suppose we have found a $u\in\mathbb{Z}\,[\omega]$ such that $|u-\sqrt{\phi}|<\epsilon$. It can be shown that $|u|$ cannot be made exactly equal to $\sqrt{\phi}$, therefore the matrix

$$\begin{pmatrix}0&-u^*\sqrt{\tau}\\u\sqrt{\tau}&0\end{pmatrix}$$

is going to be subtly non-unitary, thus we must fill the diagonal with elements $v,v^*\in\mathbb{Z}\,[\omega]$, however tiny, such that $|v|^2+|u|^2\,\tau=1$. This amounts to solving the equation $|v|^2=1-|u|^2\,\tau$ for $v\in\mathbb{Z}\,[\omega]$.

Now consider a more general case, recalling Lemma 2, where the compilation target is a Z-rotation

$$R_Z(\theta)=\begin{pmatrix}e^{-i\,\theta/2}&0\\0&e^{i\,\theta/2}\end{pmatrix}.$$

Suppose $u\in\mathbb{Z}\,[\omega]$ and $|u-e^{-i\,\theta/2}|<\epsilon$. In general, $|u|$ will be close to 1 but not exactly 1, so

$$\begin{pmatrix}u&0\\0&u^*\end{pmatrix}$$

is likely to be subtly non-unitary. Again, we need to fill the off-diagonal entries with $v\sqrt{\tau};-v^*\sqrt{\tau}, v\in\mathbb{Z}\,[\omega]$, which amounts to solving $|v|^2=(1-|u|^2)/\tau$ for $v\in\mathbb{Z}\,[\omega]$.

In order to develop a general solution for solving these norm equations, we must consider the previously defined rings $\mathbb{Z}\,[\tau]$ and $\mathbb{Z}\,[\omega]$ and automorphism $(.\,)^\bullet$.

B. Components of the norm equation

Consider the following *norm maps*, further described in Appendix A:

$$N_i : \mathbb{Z}[\omega] \to \mathbb{Z}[\tau]\,;\; N_i(\eta) = \eta\,\eta^*, \tag{12}$$
$$N_\tau : \mathbb{Z}[\tau] \to \mathbb{Z};\; N_\tau(\xi) = \xi\,\xi^\bullet, \tag{13}$$
$$N : \mathbb{Z}[\omega] \to \mathbb{Z};\; N(\eta) = N_\tau N_i(\eta), \tag{14}$$

where map (14) is called the *absolute norm* for the ring $\mathbb{Z}[\omega]$ and map (13) is called the *relative norm* in the ring extension $\mathbb{Z}[\omega]/\mathbb{Z}[\tau]$ and is shorthand for the squared complex absolute value: $N_i(x) = |x|^2$. Correctness of its interpretation as a map into the ring $\mathbb{Z}[\tau]$ stems from the fact that $\mathbb{Z}[\tau]$ is the largest real subring of $\mathbb{Z}[\omega]$ or, in other words, $\mathbb{Z}[\tau] = \mathbb{Z}[\omega] \cap \mathbb{R}$ (see Appendix A). Therefore the real-valued element $\eta\,\eta^* \in \mathbb{Z}[\tau]$.

Given a $\xi \in \mathbb{Z}[\tau]$, we introduce the relative norm equation that we want to solve for $x \in \mathbb{Z}[\omega]$:

$$N_i(x) = \xi. \tag{15}$$

The two necessary conditions for (15) to be solvable are $\xi > 0$ and $\xi^\bullet > 0$, but these conditions are in general not sufficient. We develop the complete set of conditions in the next two subsections.

C. Units and the greatest common divisor

For ring extensions such as $\mathbb{Z}[\omega]/\mathbb{Z}[\tau]$, the theory of norm equations is relatively simple, and the solutions of such equations are well understood. It is particularly simple here due to the fact that both $\mathbb{Z}[\tau]$ and $\mathbb{Z}[\omega]$ are principal ideal domains (PIDs) [20]. We describe the PID property and its consequences, and refer the reader to Appendix A for details.

First, we note that given a ring R and a principal ideal generated for an element $p \in R$, i.e., $I(p) = \{r\,p | r \in R\}$, the representation of the ideal is unique up to an invertible element of R, called a *unit*. If $u \in R$ is a unit, any element $r\,p \in I(p)$ is equal to $(r\,u^{-1})\,(u\,p)$ and thus belongs to $I(u\,p)$. It follows that the converse is also true.

For example, when $R = \mathbb{Z}$ the only units are $+1$ and -1. The ideal $I(2)$ contains both positive and negative even numbers and thus coincides with $I(-2)$. However, the group of units of $\mathbb{Z}[\tau]$ is infinite (see Lemma 15) and therefore each ideal in $\mathbb{Z}[\tau]$ has an infinite set of equivalent representations.

Second, when R is a principal ideal domain any two elements of the ring have at least one greatest common divisor. For $a, b \in R$, the ideal $I(a, b) = \{v\,a + w\,b | v, w \in R\}$ must be generated by some $g \in R$ and any common divisor of a and b divides g. Again, a greatest common divisor is only unique up to multiplicative unit of the ring R. Units of a number ring R play an important role in several of the algorithms below.

The following definition will be useful:

Definition 7. *(1) The set of of all units of a ring R is a group with respect to multiplication called the* ***unit group*** *of R and denoted by $U(R)$.*

(2) Two elements $r_1, r_2 \in R$ are ***associate*** *if there exists a unit $u \in U(R)$ such that $r_2 = u\,r_1$.*

Example 8. *(1) As per lemma 15, the unit group $U(\mathbb{Z}[\tau])$ is an infinite group generated by $\{-1, \tau\}$ and it consist of all elements $\{\pm\tau^k, k \in \mathbb{Z}\}$.*

(2) By direct computation in $\mathbb{Z}[\tau]$, $(2-\tau)^\bullet = (3+\tau) = \tau^{-2}\,(2-\tau)$. Thus $(2-\tau)^\bullet$ is associate with $(2-\tau)$.

Both rings $\mathbb{Z}[\tau]$ and $\mathbb{Z}[\omega]$ are principal ideal domains; in Section IV F an efficient algorithm for computing the greatest common divisor (GCD) in $\mathbb{Z}[\omega]$ is presented. Below, $GCD_{\mathbb{Z}[\omega]}$ refers to a greatest common divisor computed by this method or otherwise.

The goal of the following subsections is to develop an algorithm for solving the norm equation (15) and to prove that its runtime is probabilistically polynomial in the bit size of the righthand side of (15) provided a prime factorization of the righthand side is available.

D. Solvability and solutions

First we identify the *primitive element* $\theta \in \mathbb{Z}[\omega]$:

$$\theta = \omega + \omega^4 = -1 + 2\,\omega - \omega^2 + \omega^3, \tag{16}$$

where by direct computation

$$\theta^2 = \tau - 2. \tag{17}$$

Since $\tau - 2 < 0$, the value of θ in $\mathbb{C}$ is purely imaginary.

We begin by investigating Eq (15) for the special case where ξ is prime. First, consider the case when the norm of ξ is 5:

Observation 9. *If $\xi > 0, \xi^\bullet > 0$ and $N_\tau(\xi) = 5$, there exists a $k \in \mathbb{Z}$, such that $x = \pm\tau^k\,\theta$ is a solution of (15), where θ is the primitive element (16).*

Proof. Note first that $N_i(\theta) = 2-\tau$ and $N_\tau(2-\tau) = 5$. As follows from Appendix B, any other solution of $N_\tau(\xi) = 5$ is associate with either $(2-\tau)$ or $(2-\tau)^\bullet$. However, the $(2-\tau)$ element is exceptional in $\mathbb{Z}[\tau]$. It is associate with its adjoint (see Example 8). Therefore $\xi = u\,(2-\tau)$ where $u \in \mathbb{Z}[\tau]$ is a unit. Units of $\mathbb{Z}[\tau]$ are characterized in Lemma 15. For $\xi > 0, \xi^\bullet > 0$ to hold, u must be of the form $u = \tau^{2k}, k \in \mathbb{Z}$, and our observation immediately follows. □

A general solution is based on the following:

Theorem 10. *Given a prime element $\xi \in \mathbb{Z}[\tau]$, the norm equation $N_i(x) = \xi$ is solvable in $\mathbb{Z}[\omega]$ if and only if the following two conditions are satisfied:*

1. $\xi > 0, \xi^\bullet > 0$,

2. $p = N_\tau(\xi)$ *is an integer prime that is either of the form* $p = 5\,m + 1, m \in \mathbb{Z}$ *or* $p = 5$.

Assuming these conditions are satisfied, then
(a) $\tau - 2 = m^2 \bmod \xi$ *for some* $m \in \mathbb{Z}$,
(b) when $p \neq \pm 5$, *the equation (15) has at least two distinct solutions* $x = \tau^k\, s$ *and* $x^* = \tau^k\, s^*$ *for a certain* $k \in \mathbb{Z}$, *where* $s = GCD_{\mathbb{Z}[\omega]}(\xi, m - \theta)$.

The proof requires extensive algebraic number theory and is outlined in Appendix B.

Powerful general algorithms exist for solving relative norm equations in algebraic field extensions [21–23] , however Thm 10 suggests an algorithm that has probabilistically polynomial time complexity for important cases. The following methods are required:

1. an algorithm for computing square root modulo a prime,
2. an algorithm for computing GCD in $\mathbb{Z}[\omega]$,
3. an algorithm for computing $\log_\tau(unit)$ (i.e., the one that recovers the integer m given the value of τ^m).

Before proceeding, we first present a general theorem describing solutions of (15) where the righthand side is not necessarily prime.

Since $\mathbb{Z}[\tau]$ is a PID, we can factor any element into a product of a complete square and a square-free part, and then split the square-free part into prime factors. Assuming this representation, we claim the following:

Theorem 11. *For any* $\xi \in \mathbb{Z}[\tau]$ *such that* $\xi > 0, \xi^\bullet > 0$ *there exists a factorization*

$$\xi = \eta^2 \xi_1 ... \xi_r, \eta, \xi_j \in \mathbb{Z}[\tau], r \in \mathbb{Z}, r \geq 0, j = 1, ..., r \quad (18)$$

where $\xi_1, ..., \xi_r$ *are* $\mathbb{Z}[\tau]$*-primes such that* $\xi_j > 0, \xi_j^\bullet > 0, j = 1, \ldots, r$.

Given such a factorization, the norm equation $N_i(x) = \xi$ *is solvable in* $\mathbb{Z}[\omega]$ *if and only if* $p_j = N_\tau(\xi_j)$ *is an integer prime for each* $j \in \{1, ..., r\}$ *and either* $p_j = 1 \bmod 5$ *or* $p_j = 5$.

The proof is given in Appendix B.

In the context of Thm 11, we make the following:

Observation 12. $N_i(x) = \xi$ *has at least* 2^r *distinct solutions.*

Assuming that each $N_i(y) = \xi_j$ is individually solvable, each factor equation has exactly two solutions. The η^2 factor does not affect the solvability since $N_i(\eta) = \eta^2$, however, depending on the value of η, the $N_i(z) = \eta^2$ equation may have a number of other solutions besides η. In general, solving factorization (18) is as hard as factorizing an arbitrary rational integer. This part of a general solution procedure cannot be done in polynomial running time, however we may consider solving norm equations where the righthand side happens to be factorizable at a lower cost. An algorithm that solves a subclass of norm equations over $\mathbb{Z}[\omega]$ is summarized in Section IV I. We now present subalgorithms needed to implement Thm 10.

Input: $n, p \in \mathbb{Z}$, assume p is an odd prime and n is a quadratic residue $\bmod p$.
1: **procedure** TONELLI-SHANKS(n,p)
2: Represent $p - 1$ as $p - 1 = q\,2^s$, q odd.
3: **if** s=1 **then** return $\pm n^{(p+1)/4} \bmod p$;
4: **end if**
5: By randomized trial select a quadratic non-residue z , i.e. $z \in \{2, \ldots, p-1\}$ such that $z^{(p-1)/2} = -1 \bmod p$.
6: Let $c = z^q \bmod p$.
7: Let $r = n^{(q+1)/2} \bmod p$, $t = n^q \bmod p$, $m = s$.
8: **while** $t \neq 1 \bmod p$ **do**
9: By repeated squaring find the smallest $i \in \{1, \ldots, m-1\}$ such that $t^{2^i} = 1 \bmod p$
10: Let $b = c^{2^{m-i-1}} \bmod p$, $r \leftarrow r\,b$, $t \leftarrow t\,b^2$, $c \leftarrow b^2$, $m \leftarrow i$
11: **end while**
12: **end procedure**
Output: $\{r, p - r\}$.

Figure 5: Tonelli-Shanks algorithm.

E. Square root modulo prime

We present the following well-known fact (c.f., quadratic extensions in [21]) as a segway into the modular square root algorithm:

Theorem 13. *If* $p = N_\tau(\xi)$ *is an integer prime then* $\mathbb{Z}[\tau]/(\xi)$ *is effectively isomorphic to* $\mathbb{Z}_p$.

Proof. Note that $\mathbb{Z}[\tau]/(\xi)$ is a field and that $p = \xi^\bullet \xi = 0 \bmod \xi$. Therefore the natural embedding $\mathbb{Z}_p \to \mathbb{Z}[\tau]/(\xi), k \bmod p \to k \bmod \xi$ is well-defined.

Writing $\xi = a + b\tau, a, b \in \mathbb{Z}$, we prove that $b \neq 0 \bmod p$. Indeed, $p = a^2 - a\,b - b^2$. If p divides b, then p divides a^2 hence p divides a. Hence p would be proportional to p^2 which is impossible.

Thus b is invertible $\bmod p$, i.e., $\exists b_1 \in \mathbb{Z} : bb_1 = 1 \bmod p = 1 \bmod \xi$. Then $\tau = -a\,b_1 \bmod \xi$ is congruent to an integer $\bmod \xi$, hence any element of $\mathbb{Z}[\tau]$ is congruent to an integer $\bmod \xi$. Therefore the above embedding is epimorphic and in fact an isomorphism. □

In the context of Thm 13 we conclude that $\tau - 2$ is congruent to $(-a\,b_1 - 2) \bmod \xi$, and that finding an integer m such that $m^2 = \tau - 2 \bmod \xi$ is equivalent to finding a m such that $m^2 = (-a\,b_1 - 2) \bmod p$. In view of Thm 10, the existence of such m is guaranteed whenever $p = 5\,l + 1, l \in \mathbb{Z}$ or $p = 5$. In this case, computing a square root of $-a\,b_1 - 2$ modulo p is performed, constructively, using the *Tonelli-Shanks Algorithm* ([24],[25], Sec. 1.5), also given in Fig. 5. The algorithm is known to be on average probabilistically linear in bit sizes of the radicand and p, and probabilistically quadratic in the bit size of p in the worst case (c.f., [26]). For convenience, we present Thm 13 and the Tonelli-Shanks algorithm in a single procedure called SPLITTING-ROOT, given in Fig. 6.

Input: $\xi \in \mathbb{Z}[\tau]$, assume $p = N_\tau(\xi)$ is an odd prime and $p = 1 \bmod 5$.
1: **procedure** SPLITTING-ROOT($\xi = a + b\,\tau$)
2: $p \leftarrow N_\tau(\xi)$, assert $b \neq 0 \bmod p$
3: $b_1 \leftarrow b^{-1} \bmod p$
4: **return** TONELLI-SHANKS($-a\,b_1 - 2$,p)
5: **end procedure**

Figure 6: Procedure SPLITTING-ROOT: Square root of $\tau - 2$ modulo ξ.

F. Greatest common divisor in $\mathbb{Z}[\omega]$

Next we need an algorithm for computing the greatest common divisor in the $\mathbb{Z}[\omega]$ ring. We consider a "generalized binary" greatest common divisor algorithm for $\mathbb{Z}[\omega]$ that draws on ideas from [27] and implements the general method given in [28]. It requires the following:

Lemma 14. *For any element of $\eta \in \mathbb{Z}[\omega]$, either $1 + \omega$ divides η or η is associate to an element $\zeta \in \mathbb{Z}[\omega]$ such that $\zeta = 1 \bmod (1 + \omega)$.*

Proof. Any element is congruent to a rational integer $\bmod (1 + \omega)$. Since $5 = N(1 + \omega) = (1 + \omega^3)(2 - \omega + \omega^2 - \omega^3)(1 - \omega^2)(1 + \omega)$, it follows that $\eta \bmod (1 + \omega) = \eta \bmod (1 + \omega) \bmod 5$. Thus any element is congruent to one of $\{0, \pm 1, \pm 2\}$ modulo $1 + \omega$. Since -1 is a unit, any element is associate to one that is congruent to either 0, 1 or 2. It remains to note that $2 = u\,1 + (1 + \omega)$, where $u = 1 - \omega$ is a unit and therefore 2 is associate to $1 \bmod (1 + \omega)$. □

Note that the integer remainder of an element $\eta \in \mathbb{Z}[\omega]$ modulo $(1 + \omega)$ is computed by substituting -1 for ω in η . Computing the quotient and remainder of η with respect to $(1 + \omega)$ requires a total of no more than 8 integer additions. After reducing the remainder $\bmod 5$, the selection of a unit needed to associate η with the desired ζ is straightforward and immediate.

The "generalized binary" GCD algorithm based on the above Lemma is presented in Fig. 7. It follows from the analysis in [28] that this algorithm converges in a number of steps (or recursion depth) that is quadratic in bit sizes of the norms of inputs (and hence the number of steps is polylogarithmic in the magnitudes of the norms).

G. Discrete logarithm base τ

Finally, we need an algorithm for the discrete logarithm of a unit in $\mathbb{Z}[\tau]$. For completeness, we prove the following elementary lemma and then derive the desired algorithm from the proof.

Lemma 15. *(1) The group of units $U(\mathbb{Z}[\tau])$ is generated by -1 and τ.*

(2) If a unit $u \in U(\mathbb{Z}[\tau])$ is such that $u > 0, u^\bullet > 0$ then u is a perfect square in $U(\mathbb{Z}[\tau])$.

Input: $a, b \in \mathbb{Z}[\omega]$
1: **procedure** BINARY-GCD(a,b)
2: **if** one of the inputs is zero **then**
3: return the other input
4: **end if**
5: **if** $(1 + \omega)$ divides both inputs **then**
6: compute a_1: $a = (1 + \omega)a_1$; compute b_1: $b = (1 + \omega)b_1$; return $(1 + \omega)$BINARY-GCD(a_1, b_1)
7: **end if**
8: $u = v = 1$
9: **if** $(1 + \omega)$ divides neither of the inputs **then**
10: select unit u such that $u\,a = 1 \bmod (1 + \omega)$; select unit v such that $v\,b = 1 \bmod (1 + \omega)$
11: **end if**
12: let $c \in \{a, b\}$ be the input with smaller norm
Output: BINARY-GCD$(c, u\,a - v\,b)$.
13: **end procedure**

Figure 7: Procedure BINARY-GCD: GCD in $\mathbb{Z}[\omega]$.

Proof. (1) Both $-1, \tau$ are clearly units. Let $u = a + b\,\tau \in U(\mathbb{Z}[\tau])$ and define $\mu(u) = a\,b$. We show that there exists a certain $k \in \mathbb{Z}$ and $\delta = \pm 1$ such that $|\mu(u\,\delta\,\tau^k)| \leq 1$.

Since -1 is a unit we can assume w.l.o.g. that $a > 0$. Now, consider the case of $\mu(u) > 1$, implying $b > 0$. Since $N_\tau(u) = (a - b)(a + b) - a\,b = \pm 1$ and $a - b = (a\,b \pm 1)/(a + b)$ it follows that $a > b$. Consider the new unit $u' = u\,\tau = a' + b'\tau$ where $a' = b, b' = a - b$. We observe that $a' > 0, b' > 0$ and $0 < \mu(u') = a\,b - b^2 < \mu(u)$. Thus $\mu(u)$ strictly decreases when the unit is multiplied by τ but will not become < 1, as long as $\mu(u) > 1$. Therefore, there exists a positive integer k such that $\mu(u\,\tau^k) = 1$.

The case of $\mu(u) < -1$ is handled similarly, however, we repeatedly multiply the unit times τ^{-1}instead of τ. Units with $|a\,b| \leq 1$ are easily enumerated and are found to be $\{\pm 1, \pm\tau, \pm(1 - \tau) = \pm\tau^2, \pm(1 + \tau) = \pm\tau^{-1}\}$.

(2) For a unit to u to be positive, u must be of the form $u = \tau^m, m \in \mathbb{Z}$. Then $u^\bullet = (-(\tau + 1))^m$ is positive if and only if m is even. Thus $u = (\tau^{m/2})^2$. □

This proof suggests a straightforward algorithm for "decoding" a unit, called UNIT-DLOG, presented in Fig. 8.

H. The EASY-SOLVABLE predicate

The combination of Thms 10 and 11 yields a principled constructive description of solutions of a norm equation over $\mathbb{Z}[\omega]$ where the only computationally hard part is the factorization of the righthand side of the equation. A definition of what is easy to solve depends on how good we are at factorization in $\mathbb{Z}[\tau]$. We make this dependency explicit in the algorithm presented in Fig. 11 and give an example of a viable EASY-FACTOR procedure, also given in Fig. 10; as future work, further enhancements of EASY-FACTOR could lead to even better compiled circuits.

Input: unit $u = a + b\,\tau \in U(\mathbb{Z}[\tau])$
procedure UNIT-DLOG(u)
  $s \leftarrow 1$, $k \leftarrow 0$
  if $a < 0$ **then**
    $a \leftarrow -a$; $b \leftarrow -b$; $s \leftarrow -s$
  end if
  $\mu \leftarrow a\,b$
  while $|\mu| > 1$ **do**
    if $\mu > 1$ **then**
      $(a, b) \leftarrow (b, a - b)$; $k \leftarrow k - 1$
    else
      $(a, b) \leftarrow (a, a - b)$; $k \leftarrow k + 1$
    end if
    $\mu \leftarrow a\,b$
  end while ▷ $|\mu| = 1$ here
  match $v = a + b\,\tau$ with one of the $\{\pm 1, \pm\tau, \pm\tau^2, \pm\tau^{-1}\}$
  adjust s, k accordingly
  return (s, k)
end procedure
Output: (s, k) such that $s = -1, 1, k$ – integer and $u = s\tau^k$

Figure 8: Procedure UNIT-DLOG. Finds a discrete logarithm of the unit u. The procedure runtime is in $O(\log(\max\{|a|, |b|\}))$.

Procedure EASY-SOLVABLE (Fig. 9) has a single input that is assumed to be a list of factors belonging to $\mathbb{Z}[\tau]$ with their multiplicities. A factor of the form $\eta^{2s}, s \in \mathbb{Z}$, contributes a factor of η^s to the overall solution and its presence or absence does not affect the solvability of the equation. A factor of multiplicity 1 is either hard to factorize or it is prime. As per Thm 10, a prime factor $\xi \in \mathbb{Z}[\tau]$ is potentially a witness that the overall equation is not solvable, unless $p = N_\tau(\xi)$ is an integer prime and either $p = 5$ or $p = 1 \bmod 5$. We use a primality test (subroutine IS-PRIME in procedure EASY-SOLVABLE) that has probabilistically polynomial runtime and negligible probability of returning a false positive. Procedure EASY-FACTOR in Fig. 10 is an example of a minimum-effort factorizer that is sufficient for our purposes.

Input: $fl : List\langle\mathbb{Z}[\tau] \times \mathbb{Z}\rangle$
procedure EASY-SOLVABLE(fl)
  for $i \in \{0..length(fl) - 1\}$ **do**
    match $fl[i]$ with $(\xi, k), \xi \in \mathbb{Z}[\tau], k \in \mathbb{Z}$
    if $k = 1 \bmod 2$ **then**
      if $\xi \neq 5$ **then**
        $p \leftarrow N_\tau(\xi)$
        $r \leftarrow p \bmod 5$
        if not IS-PRIME(p) or $r \notin \{0, 1\}$ **then**
          return FALSE
        end if
      end if
    end if
  end for
  return TRUE
end procedure
Output: TRUE if fl is a factorization of an easy and solvable instance of the equation.

Figure 9: Procedure EASY-SOLVABLE: checks if the given instance of the norm equation can be solved in polynomial time.

Input: $\xi \in \mathbb{Z}[\tau]$
procedure EASY-FACTOR($\xi = a + b\,\tau, a, b \in \mathbb{Z}$)
  $c \leftarrow GCD(a, b)$; $a_1 \leftarrow a/c$; $b_1 \leftarrow b/c$; $\xi_1 \leftarrow a_1 + b_1\,\tau$
  if $c = d^2, d \in \mathbb{Z}$ **then**
    $ret \leftarrow List((d, 2))$
  else
    if $c = 5\,d^2, d \in \mathbb{Z}$ **then**
      $ret \leftarrow List((d, 2), (5, 1))$
    else
      return $List((\xi, 1))$
        ▷ equation is not going to be solvable
    end if
  end if
  $n \leftarrow N_\tau(\xi_1)$
  if $n = 0 \bmod 5$ **then**
    $\xi_2 \leftarrow \xi_1/(2 - \tau)$
    return $ret + ((2 - \tau), 1) + (\xi_2, 1)$
  else
    return $ret + (\xi_1, 1)$
  end if
end procedure
Output: Returns lightweight factorization of input.

Figure 10: Procedure EASY-FACTOR: a minimum-effort factorizer.

I. The algorithm

Using the described procedures, we present an algorithm for solving norm equations, SOLVE-NORM-EQUATION, given in Fig. 11. The algorithm first checks the necessary conditions $\xi > 0, \xi^\bullet > 0$ on the righthand side of the equation, and provided the conditions are satisfied, invokes EASY-FACTOR to preprocess ξ. If the resulting list of factors is EASY-SOLVABLE, we consider each factor to either have even multiplicity or be a power of an allowed prime in $\mathbb{Z}[\tau]$. An allowed prime is either 5 or $2 - \tau$ with norm 5, or some other prime with norm p such that $p = 1 \bmod 5$. 5 is equal to the norm of $2\,\tau + 1$. In the case of $2 - \tau$, we exploit the identity $|\omega + \omega^4|^2 = 2 - \tau$ and induce the factor $(\omega + \omega^4)$ into the solution. In the more general case we need to perform all the steps prescribed by Thm 10, specifically: (1) represent $\tau - 2$ as a square of integer M modulo ξ (as in Fig. 6), (2) compute the GCD y of ξ and $M - (\omega + \omega^4)$ in $\mathbb{Z}[\omega]$ (as in Fig. 7) , (3) obtain a unit u that associates ξ with $|y|^2$, (4) represent the unit u as τ^m using the UNIT-DLOG procedure (as in Fig. 8). Having performed these steps, we a obtain a factor of the desired solution corresponding to the allowed prime factor of the righthand side. The algorithm terminates when all the factors of the righthand side have been inspected.

Input: $\xi \in \mathbb{Z}[\tau]$

procedure SOLVE-NORM-EQUATION(ξ)
  if $\xi < 0$ or $\xi^\bullet < 0$ **then**
    return UNSOLVED
  end if
  $fl \leftarrow$EASY-FACTOR(ξ)
  if not EASY-SOLVABLE(fl) **then**
    return UNSOLVED
  end if
  $x \leftarrow 1$
  for $i \in \{0..length(fl)-1\}$ **do**
    match $fl[i]$ with $(\xi_i, m), \xi_i \in \mathbb{Z}[\tau], m \in \mathbb{Z}$
    $x \leftarrow x\,\xi_i^{m/2}$
    if $m = 1 \bmod 2$ **then** ▷ assert ξ_i is easy factor
      if $\xi_i = 5$ **then**
        $x \leftarrow x\,(2\tau+1)$
      else
        if $\xi_i = 2-\tau$ **then**
          $x \leftarrow x\,(\omega+\omega^4)$
        else
          $M \leftarrow$ SPLITTING-ROOT(ξ_i)
            ▷ $M^2 = \tau - 2 \bmod \xi_i$
          $y \leftarrow$ BINARY-GCD($\xi_i, M-(\omega+\omega^4)$)
            ▷ $(\omega+\omega^4)^2 = \tau - 2$
          $u \leftarrow \xi_i/|y|^2$
            ▷ u – unit, $u > 0$, $u^\bullet > 0$
          $(s, m) \leftarrow$ UNIT-DLOG(u)
            ▷ $s = 1$, m – even
          $x \leftarrow x\,\tau^{m/2} y$
        end if
      end if
    end if
  end for
  return x
end procedure

Output: x from $\mathbb{Z}[\omega]$ such that $|x|^2 = \xi$

Figure 11: Procedure SOLVE-NORM-EQUATION: finds a solution to an "easy" instance of the norm equation in probabilistic polynomial time.

Example 16. *Consider the relative norm equation*

$$N_i(x) = \xi = 760 - 780\,\tau. \tag{19}$$

This equation turns out to be EASY-SOLVABLE with one of the solutions

$$x = 2\,(4+3\,\tau)(12 - 20\,\omega + 15\,\omega^2 - 3\,\omega^3). \tag{20}$$

Following the EASY-FACTOR procedure, it is relatively easy to obtain the following list of factors for ξ:

$$fl = \{(2,2), (5,1), ((2-\tau),1), ((15-8\,\tau),1)\}.$$

All but the last factor in this list yield easy partial solutions: $N_i(2) = 2^2$, $N_i(2\,\tau+1) = 5$,$N_i(\omega+\omega^4) = 2-\tau$. For the last factor, we find that $p = N_\tau(15-8\,\tau) = 281$ is prime and $p = 1 \bmod 5$. SPLITTING-ROOT$(15-8\,\tau)$ yields 63 and BINARY-GCD$(15-8\,\tau,\ 63-(\omega+\omega^4))$ yields $y = 3 + 2\,\omega - 7\,\omega^2 + 7\,\omega^3$. By direct computation, $(15-8\,\tau)/|y|^2 = 5 + 3\,\tau = \tau^{-4}$ and thus $N_i(\tau^{-2}\,(3 + 2\,\omega - 7\,\omega^2 + 7\,\omega^3)) = 15 - 8\,\tau$. The value in (20) is a routine simplification of $2\,(2\,\tau+1)\,\tau^{-2}(\omega + \omega^4)(3 + 2\,\omega - 7\,\omega^2 + 7\,\omega^3)$. (Note that further simplification is possible using $\tau = \omega - \omega^3$ and is left as an exercise.)

V. APPROXIMATION

In this section we combine methods developed in previous sections and describe an algorithm for approximating unitaries of the form $R_z(\phi)X$ and $R_z(\phi)X$ with $\langle \mathcal{F}, \mathcal{T} \rangle$-circuits (or equivalently, $\langle \sigma_1, \sigma_2 \rangle$-circuits). Recall that we measure the quality of approximation ε using the global phase-invariant distance

$$d(U,V) = \sqrt{1 - |tr(UV^\dagger)|\,/2}. \tag{21}$$

The quality of approximation ε defines the problem size. Our algorithm produces circuits of length $O(\log(1/\varepsilon))$ which meets the asymptotic worst-case lower bound [29] for such a circuit. The algorithm is probabilistic in nature and on average requires running time in $O(\log^c(1/\varepsilon))$ to find an approximation where c is a constant close to but smaller than 2.0, according to our empirical estimates .

We first discuss all details of the algorithm for approximating $R_z(\phi)$ and then show how the same tools allow us to find approximations of $R_z(\phi)X$.

There are two main stages in our algorithm: the first stage approximates $R_z(\phi)$ with an exact unitary $U[u, v, 0]$ and then uses the exact synthesis algorithm (Figure 4) to find a circuit implementing $U[u, v, 0]$. The second stage is completely described in Section III; here we focus on the first stage.

The expression for the quality of approximation in the first case simplifies to

$$d(R_z(\phi), U[u,v,0]) = \sqrt{1 - \left|\mathrm{Re}(ue^{i\phi/2})\right|}$$

We see that the quality of approximation depends only on u, the top left entry of $U[u, v, 0]$. Therefore, to solve the first part of the problem it is sufficient to find u from $\mathbb{Z}[\omega]$ such that $\sqrt{1 - \left|\mathrm{Re}(ue^{i\phi/2})\right|} \le \varepsilon$. However, there is an additional constraint that must be satisfied: there must exist a v from $\mathbb{Z}[\omega]$ such that $U[u, v, 0]$ is unitary, in other words the following equation must be solvable:

$$|v|^2 = \xi, \text{ for } \xi = \varphi(1 - |u|^2). \tag{22}$$

This is precisely the equation studied in Section IV. As discussed, in general the problem of deciding whether such a v exists and then finding it is hard. It turns out, however, that in our case there is enough freedom to pick (find) "easy" instances and obtain a solution in polynomial time without sacrificing too much quality.

There is an analogy to this situation: it is well known that factoring a natural number into prime factors is a hard problem. However, checking that the number is

Input: ϕ – defines $R_z(\phi)$, ε – precision
$C \leftarrow \sqrt{\varphi/4}$
$m \leftarrow \lceil \log_\tau(C\varepsilon) \rceil + 1$
Find k such that $\theta = -\phi/2 - \pi k/5 \in [0, \pi/5]$
not-found $\leftarrow$ true, $u \leftarrow 0, v \leftarrow 0$
while *not-found* **do**
  $u_0 \leftarrow$ RANDOM-SAMPLE$(\theta, \varepsilon, 1)$ ▷ See Figure 13
  $\xi \leftarrow \varphi\left(\varphi^{2m} - |u_0|^2\right)$
  $fl \leftarrow$ EASY-FACTOR(ξ)
  if EASY-SOLVABLE(fl) **then**
    not-found $\leftarrow$ false
    $u \leftarrow \omega^k \tau^m u_0$
    $v \leftarrow \tau^m$SOLVE-NORM-EQUATION(ξ)
  end if
end while
$C \leftarrow$ EXACT-SYNTHESIZE$(U[u, v, 0])$
Output: Circuit C such that $d(C, R_z(\phi)) \le \varepsilon$

Figure 12: The algorithm for approximating $R_z(\phi)$ by an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit with $O(\log(1/\varepsilon))$ gates and precision at most ε. Runtime is probabilistically polynomial as a function of $\log(1/\varepsilon)$.

prime can be done in polynomial time. In other words, given a natural number N one can efficiently decide if it is an easy instance for factoring. Now imagine the following game: one is given a uniformly chosen random number from the interval $[0, N]$ and one wins each time they can factor it. How good is this game? The key here is the Prime Number Theorem. It states that there are $\Theta(N/\log(N))$ primes in the interval $[0, N]$. Therefore one can win the game with probability at least $\Omega(1/\log(N))$. In other words, the number of trials one needs to make before winning scales as $O(\log(N))$. In our case the situation is somewhat similar and N is of order $1/\varepsilon$.

At a high level, during our approximation procedure (Figure 12) we perform a number of trials. During each trial we first randomly pick a u from $\mathbb{Z}[\omega]$ that achieves precision ε and then check that the instance of the norm equation can be easily solved. Once we find such an instance we compute v and construct a unitary $U[u, v, 0]$.

We generate a random element u from $\mathbb{Z}[\omega]$ that has the desired precision using procedure RANDOM-SAMPLE. To achieve a better constant factor in front of $\log(1/\varepsilon)$ for the length of the circuit we randomly chose $u_0 = u\varphi^m$ instead of u. It is easy to recover u as $\tau = \varphi^{-1}$ and $u = u_0\tau^n$.

In Figure 14, when $r = 1$, the light gray circular segment corresponds to such u_0 that $U[u, v, 0]$ is within ε from $R_z(\phi)$. The element u_0 is a complex number and, as usual, the x-axis of the plot corresponds to the real part and the y-axis to the imaginary part. All random samples that we generate belong to the dark gray parallelogram and have the form $a_x + b_x\tau + i\sqrt{2-\tau}(a_y + b_y\tau)$ (note that $i\sqrt{2-\tau}$ is equal to $\omega + \omega^4$ and belongs to $\mathbb{Z}[\omega]$). We first randomly choose an imaginary part and then a real part. To find an imaginary part we randomly choose a real number y and then approximate it with $\sqrt{2-\tau}(a_y + b_y\tau)$

Input: θ – angle between 0 and $\pi/5$, ε – precision, $r \ge 1$
procedure RANDOM-SAMPLE(θ, ε, r) ▷ See Fig. 14
  $C \leftarrow \sqrt{\varphi/(4r)}$
  $m \leftarrow \lceil \log_\tau(C\varepsilon r) \rceil + 1$
  $N \leftarrow \lceil \varphi^m \rceil$
  $y_{min} \leftarrow r\varphi^m(\sin(\theta) - \varepsilon\left(\sqrt{4-\varepsilon^2}\cos(\theta) + \varepsilon\sin(\theta)\right)/2)$
  $y_{max} \leftarrow r\varphi^m(\sin(\theta) + \varepsilon\left(\sqrt{4-\varepsilon^2}\cos(\theta) - \varepsilon\sin(\theta)\right)/2)$
  $x_{max} \leftarrow r\varphi^m((1-\varepsilon^2/2)\cos(\theta) - \varepsilon\sqrt{1-\varepsilon^2/4}\sin(\theta))$
  $x_c \leftarrow x_{max} - r\varepsilon^2\varphi^m/(4\cos(\theta))$
  Pick random integer j from $[1, N-1]$
  $y \leftarrow y_{min} + j(y_{max} - y_{min})/N$
  $a_y + \tau b_y \leftarrow$ APPROX-REAL$(y/\sqrt{2-\tau}, m)$ ▷ Fig. 15
  $x \leftarrow x_c - ((a_y + b_y\tau)\sqrt{2-\tau} - y_{min})\tan(\theta)$
  $a_x + \tau b_x \leftarrow$ APPROX-REAL(x, m) ▷ Fig. 15
  return $a_x + \tau b_x + \sqrt{\tau-2}(a_y + \tau b_y)$
end procedure

Figure 13: The algorithm for picking a random element of $\mathbb{Z}[\omega]$ that is in the dark gray region in Figure 14. Number of different outputs of the algorithm is in $O(1/\varepsilon)$.

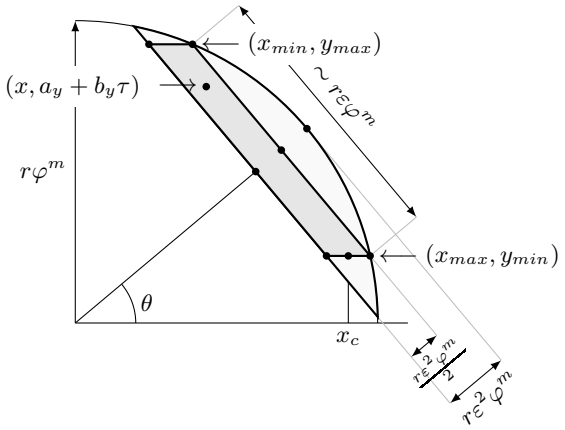

Figure 14: An ε-region and visualization of variables used in RANDOM-SAMPLE procedure (Figure 13).

using the APPROX-REAL (Figure 15) procedure. Once we find $\sqrt{2-\tau}(a_y + b_y\tau)$, we choose the x-coordinate as shown in Figure 14 and approximate it with $a_x + b_x\tau$.

The problem of approximating real numbers with numbers of the form $a + bz$ for integers a, b and irrational z is well studied. The main tool is continued fractions. It is also well known that Fibonacci numbers $\{F_n\}$,

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}, n \ge 2,$$

are closely related to the continued fraction of the Golden section φ and its inverse τ. The correctness of procedure APPROX-REAL (Figure 15) is based on the following very well-known result connecting τ and the Fibonacci numbers; we state it in a convenient form and provide the proof for completeness.

Input: x – real number, n – defines precision as $\tau^{n-1}(1-\tau^n)$

procedure APPROX-REAL(x,n)
  $p \leftarrow F_n, q \leftarrow F_{n+1}$ ▷ F_n – Fibonacci numbers
  $u \leftarrow (-1)^{n+1}F_n, v \leftarrow (-1)^n F_{n-1}$ ▷ $up + vq = 1$
  $c \leftarrow \lfloor xq \rceil$
  $a \leftarrow cv + p\lfloor cu/q \rceil$
  $b \leftarrow cu - q\lfloor cu/q \rceil$
  return $a + \tau b$
end procedure

Output: $a + b\tau$ s.t. $|x - (a + b\tau)| \le \tau^{n-1}(1 - \tau^n), |b| \le \varphi^n$

Figure 15: The algorithm for finding integers a, b such that $a + b\tau$ approximates the real number x with precision $\tau^{n-1}(1 - \tau^n)$

Proposition 17. *For any integer n*

$$\left|\tau - \frac{F_n}{F_{n+1}}\right| \le \frac{\tau^n}{F_{n+1}}$$

and $F_n \ge (\varphi^n - 1)/\sqrt{5}$.

Proof. First we define the family of functions $\{f_n\}$ such that $f_1(x) = x$ and $f_n(x) = 1/(1 + f_{n-1}(x))$ for $n \ge 2$. It is not difficult to check that $f_n(\tau) = \tau$. It can be shown by induction on n that $f_n(1)$ is equal to F_n/F_{n+1}. Therefore, to prove the first part of the proposition we need to show that

$$|f_n(\tau) - f_n(1)| \le \tau^n/F_{n+1}.$$

We proceed by induction. The statement is true for n equal to 1. Using the definition of f_n it is not difficult to show

$$|f_{n+1}(\tau) - f_{n+1}(1)| = f_{n+1}(\tau)f_{n+1}(1)\,|f_n(\tau) - f_n(1)|\,.$$

Using $f_{n+1} = \tau$ and the inequality for $|f_n(\tau) - f_n(1)|$ we complete the proof of the first part.

To show the second part it is enough to use the well-known closed form expression for Fibonacci numbers $F_n = (\varphi^n - (-\tau)^n)/\sqrt{5}$. □

At a high level, in procedure APPROX-REAL we approximate τ with a rational number p/q and find the approximation of x by a rational of the form $a + bp/q$. To get good resulting precision we need to ensure that $b(\tau - p/q)$ is small, therefore we pick b in such a way that $|b| \le q/2$. The details are as follows.

Lemma 18. *Procedure APPROX-REAL (Figure 15) outputs integers a, b such that $|x - (a + b\tau)| \le \tau^{n-1}(1 - \tau^n)$, $|b| \le \varphi^n$ and terminates in time polynomial in n.*

Proof. First, identity $F_{n+1}F_{n-1} - F_n^2 = (-1)^n$ for Fibonacci numbers implies $uv + pq = 1$. Next, by choice of c we have $|xq - c| \le 1/2$, therefore $|x - c/q| \le 1/2q$. By Proposition 17, $1/2q$ is less than $\tau^n\sqrt{5}(1 - \tau^n)/2$; it remains to show that c/q is within distance $\tau^n/2$ from $a + b\tau$ by the triangle inequality. By choice of a, b we have $c = aq + bp$ and

$$|c/q - (a + b\tau)| = |b|\,|\tau - p/q|\,.$$

Using Proposition 17, equality $q = F_{n+1}$ and inequality $|b| \le q/2$ we conclude that $|c/q - (a + b\tau)| \le \tau^n/2$.

The complexity of computing n^{th} Fibonacci number is polynomial in n. Assuming that the number of bits used to represent x is proportional to n all arithmetic operations also have complexity polynomial in n. □

There are two main details of the RANDOM-SAMPLE procedure we need to clarify. The first is that the result is indeed inside the dark gray parallelogram on Figure 14. This is achieved by picking real values x, y far enough from the border of the parallelogram and then choosing the precision parameter m for APPROX-REAL in such a way that $a_x + b_x\tau$(close to x) and $a_y + b_y\tau$(close to $y/\sqrt{2 - t}$) stay inside the parallelogram.

The second important detail is the size of resulting coefficients a_x, b_x, a_y, b_y. It is closely related to the number of gates in the resulting circuit. Therefore it is important to establish an upper bound on coefficients size. The following lemma provides a rigorous summary:

Lemma 19. *When the third input $r \ge 1$, procedure RANDOM-SAMPLE has the following properties:*

- *there are $O(1/\varepsilon)$ different outputs and each of them occurs with the same probability,*
- *the procedure outputs an element u_0 of $\mathbb{Z}[\omega]$ from the dark gray parallelogram P in Figure 14,*
- *the Gauss complexity measure of u_0 is in $O(1/\varepsilon)$.*

Proof. By construction, the algorithm produces $N - 1$ outputs with equal probability. It is not difficult to check that N is in $O(1/\varepsilon)$. We first show that the outputs are all distinct and their y coordinate is in $[y_{min}, y_{max}]$. This follows from an estimate

$$|y - (a_y + b_y\tau)|\,\sqrt{2 - \tau} \le (y_{max} - y_{min})/2N$$

because each randomly generated y is at least distance $(y_{max} - y_{min})/N$ from any other randomly generated y and also y_{min}, y_{max}. To show the estimate we use the result of Lemma 18 and check that

$$\tau^{m-1}(1 - \tau^m)\sqrt{2 - \tau} \le (y_{max} - y_{min})/2N$$

which is straightforward, but tedious. If we concentrate only on terms that are first order in ε we get:

$$\tau^{m-1}(1 - \tau^m) \lesssim C\varepsilon r, (y_{max} - y_{min})/2N \gtrsim \varepsilon\cos(\theta)r. \tag{23}$$

The constraint on θ gives $\cos(\theta) \ge \varphi/2$. From $C\sqrt{2 - \tau} < \varphi/2$ we conclude that the inequality is true for the terms that are first order in ε.

To show that the procedure output belongs to the parallelogram P, it is sufficient to check that $a_x + b_x\tau$

is within distance $\varphi^m r\varepsilon^2/4$ (half of the parallelogram height) from $x_c - (a_y + b_y\tau - y_{min})\sin(\theta)$. Again using Lemma 18, it is sufficient to show that

$$\tau^{m-1}(1-\tau^m) \leq \varphi^m r\varepsilon^2/4.$$

We again analyze the expression up to the first order terms in ε. We note that $\varphi^m r\varepsilon^2/4 \eqsim \varepsilon/(4C\tau)$; combining it with the inequality above, using (23) and $C = \sqrt{\varphi/(4r)}$, we conclude that all outputs of the algorithm are inside parallelogram P.

To show the last property, we note that by Lemma 18 both $|b_x|, |b_y|$ are bounded by φ^m. The same is true for $|a_x|, |a_y|$. Indeed, $|x|, |y|$ are both of order φ^m and

$$|a_y| \lesssim |y/\sqrt{2-\tau} - b_x\tau| + C\varepsilon r,\ |x_y| \lesssim |x - b_x\tau| + C\varepsilon r.$$

This implies that if we write u_0 as $\sum_k \alpha_k\omega^k$ each integer α_k will be of order φ^m which is the same as $O(1/\varepsilon)$. Using the upper bound on the Gauss complexity measure $G(u_0)$ in terms of α_k from Proposition 4 we conclude that $G(u_0)$ is in $O(1/\varepsilon)$. □

The technical tools that we have developed so far are sufficient to verify that our approximation algorithm achieves the required precision and produces circuits of length $O(1/\log(1/\varepsilon))$. The remaining part is to show that on average the algorithm requires $O(\log^c(1/\varepsilon))$ steps. It relies on the following conjecture, similar in nature to the Prime Number Theorem:

Conjecture 20. *Let $\pi(M)$ be the number of elements ξ from $\mathbb{Z}[\tau]$ such that $N_\tau(\xi)$ is a prime representable as $5n+1$ and less than M, then $\pi(M)$ is in $\Theta(M/\log(M))$.*

The conjecture defines the frequency of easy instances of the norm equation during the sampling process. Finally we prove the main theorem.

Theorem 21. *Approximation algorithm (Figure 12) outputs a $\langle\mathcal{F},\mathcal{T}\rangle$-circuit C of length $O(\log(1/\varepsilon))$ such that $d(C, R_z(\phi)) \leq \varepsilon$. On average the algorithm runtime is in $O(\log^c(1/\varepsilon))$ if Conjecture 20 is true.*

Proof. First we show that the algorithm achieves the desired precision and produces a $\langle\mathcal{F},\mathcal{T}\rangle$-circuit of length $O(\log(1/\varepsilon))$. Both statements follow from Lemma 19. It is not difficult to check that the light gray segment on Figure 14 defines all u_0 such that $d(U[u_0\tau^m\omega^k, v, 0], R_z(\phi))$ is less than ε. Therefore, by picking samples from the dark gray parallelogram P, we ensure that we achieve precision ε. The value of the Gauss complexity measure is in $O(1/\varepsilon)$, therefore by Theorem 6 the length of the resulting circuit is in $O(1/\log(\varepsilon))$.

There are two necessary conditions for predicate EASY-SOLVABLE to be true:

(1) $N_\tau(\xi)$ is prime and equals $5n+1$ for integer n,

(2) $\xi > 0, \xi^\bullet > 0$.

Let p_M be the probability that the first condition is true when we choose ξ uniformly at random and $N_\tau(\xi)$ is bounded by M. Following Conjecture 20, we assume that p_M is in $O(1/\log(M))$. In our case M is of order φ^{2m} and therefore the probability of getting an instance solvable in polynomial time is in $O(1/\log(1/\varepsilon))$.

Now we show that the second condition is satisfied by construction. Part $\xi > 0$ is trivial because procedure RANDOM-SAMPLE always generates u_0 such that $|u_0| \leq \varphi^m$. For the second part we use Proposition 4 and note that for non-zero $u_0\tau^n$ the value of the Gauss complexity measure is

$$G(u_0\tau^n) = |(u_0\tau^n)^\bullet|^2 + |u_0\tau^n|^2 \geq 2.$$

We conclude that $|(u_0\tau^n)^\bullet|^2 \geq 1$ which gives

$$\xi^\bullet = \tau^{2m+1}(|(u_0\tau^n)^\bullet|^2 - 1) \geq 0$$

as required.

In summary, checking that an instance of ξ is easily solvable can be done in time polynomial in $\log(1/\varepsilon)$ using, for example, Miller-Rabin Primality Test, the average number of loop iterations is in $O(\log(1/\varepsilon))$, an instance of the norm equation when ξ is prime can be solved in time that is on average is in $O(\log^d(1/\varepsilon))$ for some positive d. We conclude that on average the algorithm runs in time $O(\log^c(1/\varepsilon))$ for some positive constant c. □

Input: ϕ – defines $R_z(\phi)X$, ε – precision
1: $r \leftarrow \sqrt{\varphi}, C \leftarrow \sqrt{\varphi/(4r)}$
2: $m \leftarrow \lceil \log_\tau(C\varepsilon r)\rceil + 1$
3: Find k such that $\theta = \phi/2 + \pi/2 - \pi k/5 \in [0, \pi/5]$
4: *not-found* $\leftarrow$ true, $u \leftarrow 0, v \leftarrow 0$
5: **while** *not-found* **do**
6: $u_0 \leftarrow$ RANDOM-SAMPLE(θ, ε, r) ▷ See Figure 13
7: $\xi \leftarrow \varphi^{2m} - \tau|u_0|^2$
8: $fl \leftarrow$ EASY-FACTOR(ξ)
9: **if** EASY-SOLVABLE(fl) **then**
10: *not-found* $\leftarrow$ false
11: $v \leftarrow \omega^k\tau^m u_0$
12: $u \leftarrow \tau^m$SOLVE-NORM-EQUATION(ξ)
13: **end if**
14: **end while**
15: $C \leftarrow$ EXACT-SYNTHESIZE$(U[u, v, 0])$
Output: Circuit C such that $d(C, R_z(\phi)X) \leq \varepsilon$

Figure 16: The algorithm for approximating $R_z(\phi)X$ by an $\langle\mathcal{F},\mathcal{T}\rangle$-circuit with $O(\log(1/\varepsilon))$ gates and precision at most ε. Runtime is probabilistic polynomial as a function of $\log(1/\varepsilon)$.

The algorithm for approximating $R_z(\phi)X$ (Figure 16) can now be easily constructed based on ideas discussed above. First we simplify the expression for the distance

$$d(U[u, v, 0], R_z(\phi)X) = \sqrt{1 - \sqrt{\tau}\left|\mathrm{Re}(ve^{-i(\phi/2+\pi/2)})\right|}$$

and notice that in this case it depends only on the bottom left entry of the unitary $U[u, v, 0]$. Now u and v have opposite roles in comparison to the algorithm for approximating $R_z(\phi)$. Again, to get a better constant factor in

front of $\log(1/\varepsilon)$ in the circuit size, we randomly pick v_0 such that $d(U[u, \varphi^m v_0, 0], R_z(\phi)X) \leq \varepsilon$. We use procedure RANDOM-SAMPLE to generate random v_0. When calling the procedure, we set the third input parameter r to $\sqrt{\varphi}$ to take into account that bottom left entries of exact-unitaries are rescaled by factor $\sqrt{\tau}$. Once we picked v_0 we check that there exist an exact unitary with bottom right entry $v = \tau^m v_0 \sqrt{\tau}$. In other words we solve norm equation

$$|u|^2 = \xi \text{ for } \xi = 1 - \tau|v|^2.$$

The necessary condition $\xi^\bullet \geq 0$ is always satisfied because $1 + \varphi|v^\bullet|^2$ is always positive. Once we find an "easy" instance of the norm equation we solve it and construct an exact unitary that gives the desired approximation. Our result regarding the approximation algorithm for $R_z(\phi)X$ is summarized by the following theorem.

Theorem 22. *Approximation algorithm (Figure 16) outputs a $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit C of length $O(\log(1/\varepsilon))$ such that $d(C, R_z(\phi)X) \leq \varepsilon$. On average the algorithm runtime is in $O(\log^c(1/\varepsilon))$ if Conjecture 20 is true.*

The proof is completely analogous to the proof of Theorem 21 and we do not present it here.

VI. EXPERIMENTAL RESULTS

In this section we evaluate the approximation quality of our algorithm as a function of $\langle \sigma_1, \sigma_2 \rangle$-circuit size (depth) and the algorithm runtime. We not only confirm the results established in previous sections, but also show that constants hidden in the big-O notation are quite reasonable, making our algorithm useful in practice.

We experiment over several input sets of input unitaries and precisions, as summarized in Table I. Each experiment is performed similarly. First, we request an approximation of a set of unitaries for certain precisions. In some experiments, we run the algorithm for the same unitary and precision several times to see the influence of the probabilistic nature of the algorithm on the result. Next, we aggregate collected data for a given precision by taking the mean, min or max of the parameter of interest over the set of all unitaries considered in the experiment. We compare to a Brute Force Search algorithm, from which we request approximation of the set of unitaries, that outputs the best precision that can be achieved using at most N σ gates for each input angle. The largest N for our database is 25. In this case we aggregate collected data for a given N.

We implemented our algorithm using C++. There are two third-party libraries used: PARI/GP[23] which provides a relative norm equation solver and primality test, and *boost::multiprecision* which includes high-precision integer and floating-point types. All experimental results described in this section were obtained on a computer with Intel Core i7-2600 (3.40GHz) processor and 8 GB of RAM. Our implementation does not use any parallelism.

A. Quality Evaluation

We evaluate the approximation quality of our algorithm on four large sets of inputs. Two of them are used to evaluate the approximation quality of $R_z(\phi)$ rotations and the other two for $R_z(\phi)X$. For both rotation types, one set covers uniformly the range of angles $[0, 2\pi]$ (U-RZ-BIG, U-RZX-BIG) and the other one includes rotations that are particularly important in applications. For $R_z(\phi)$ rotations, we study angles ϕ of the form $\frac{\pi}{2^n}$ used in the Quantum Fourier Transform (input set RZ-HIGH); for $R_z(\phi)X$, we look at the quality of the Pauli X gate approximation (input set X-HIGH). The results are presented in Figure 17. In addition to the average number of gates, we show minimal and maximal number of gates needed to achieve the required precision, which demonstrates the stability of the quality of our algorithm.

One of the baselines we compare the quality of our algorithm to is Brute Force Search. We built a database of optimal $\langle \sigma_1, \sigma_2 \rangle$-circuits with up to 25 gates and used it to find optimal approximations of unitaries from datasets U-RZ and U-RZX. The highest average precision that we were able to achieve is around $10^{-2.5}$. We evaluated our number theoretic algorithm on the same range of precisions; the results are presented in Figure 18. For our algorithm, the average coefficient in front of $\log_{10}(1/\varepsilon)$ is only 18% larger than the average for the optimal approximations of $R_z(\phi)$ and 40% larger for $R_z(\phi)X$.

Figure 19b shows the exponential scaling of the number of optimal circuits with a given number of $\langle \sigma_1, \sigma_2 \rangle$ gates and confirms that the Brute Force Search becomes infeasible exponentially quickly. In summary, our algorithm finds circuits for unitaries $R_z(\phi)$ and $R_z(\phi)X$ exponentially faster than Brute Force Search and with very moderate overhead.

The previous state-of-the-art method for solving the unitary approximation problem for the Fibonacci braid basis in polynomial time is the Solovay-Kitaev algorithm. The Solovay-Kitaev algorithm can be applied to any gate set and does not take into account the number theoretic structure of the approximation problem. Here we provide a rough estimate of its performance when approximating using $\langle \sigma_1, \sigma_2 \rangle$-circuits.

We consider approximation using special unitaries in this case. The version of the Solovay-Kitaev algorithm described in [11] boosts the quality of the approximation provided by a fixed-size epsilon net. It is crucial for the overall estimate to find the quality provided by an epsilon net depending on the maximal size of the optimal circuit in it. We use the scaling of the size of our database (Figure 19b) of optimal circuits and a rough volume argument to get the estimate.

Consider the problem of approximating states, which is the same as approximating single-qubit special unitaries.

	Unitaries			Precisions			Runs per
Name	formula	k_{min}	k_{max}	formula	k_{min}	k_{max}	unitary
U-RZ-SMALL	$R_z(\frac{2\pi k}{10^3})$	1	$1 \cdot 10^3$	10^{-k}	2	14	1
U-RZ-BIG	$R_z(\frac{2\pi k}{4 \cdot 10^3})$	1	$4 \cdot 10^3$	10^{-k}	2	30	1
RZ-HIGH	$R_z(\frac{\pi}{2^k})$	2	$6 \cdot 10^1$	10^{-k}	2	100	10
U-RZ-LOW	$R_z(\frac{2\pi k}{10^3})$	1	$1 \cdot 10^3$	$10^{-k/8}$	8	12	1
U-RZ	$R_z(\frac{\pi k}{10^4})$	1	$1 \cdot 10^4$	—	—	—	—
U-RZX-BIG	$R_z(\frac{2\pi k}{4 \cdot 10^3})X$	1	$4 \cdot 10^3$	10^{-k}	2	30	1
U-RZX-LOW	$R_z(\frac{2\pi k}{10^3})X$	1	$1 \cdot 10^3$	$10^{-k/8}$	8	12	1
U-RZX	$R_z(\frac{\pi k}{10^4})X$	1	$1 \cdot 10^4$	—	—	—	—
X-HIGH	X	—	—	10^{-k}	2	100	500

Table I: Sets of inputs used for the experiments.

Our ε net should cover the Bloch sphere with overall surface area 4π. Each state will cover roughly an area of the sphere equal to $\pi\varepsilon^2$. Therefore, for n being the size of the longest circuit:

$$\pi\varepsilon^2 10^{(0.275x+0.592)}/2 \simeq 4\pi.$$

We divide the estimate for the number of unitaries by two to get the estimate for the number of states. This is because there are only up to global phase two distinct exact unitaries of the form $U[x, y, k]$ for given x, y (for k=0 and k=1). Other values of k can be reduced to 0 or 1 using the identity $\omega^s U[x, y, k] = U[x\omega^s, y\omega^s, k + 2s]$. We also assume that $U[x, y, k]$ and $U[x\omega^s, y\omega^s, k]$ have very similar cost.

The database we are using in other experiments includes circuits with up to 25 gates and requires around 5GB of RAM to be built. In our estimate for the performance of the Solovay-Kitaev algorithm we assume that with enough engineering effort one can build a database with up to 30 gates. Our estimates result in the following:

$$\log_{10}(1/\varepsilon_n) \simeq 0.137n - 0.155, \log_{10}(1/\varepsilon_{30}) \simeq 3.97$$
$$n(\log_{10}(1/\varepsilon_n)) \simeq 7.27\varepsilon + 1.127.$$

The estimate is more optimistic in comparison to results of our Brute Force Search as now we consider the full special unitary group instead of its subsets $R_z(\phi)$ and $R_z(\phi)X$. With these numbers in hand, we use the analysis of the Solovay-Kitaev algorithm in [11].

Figure 19 compares our estimates for the size of circuits produced by the Solovay-Kitaev algorithm and the sizes from running our algorithm. In particular, for precision 10^{-10} our algorithm produces twenty times smaller circuits and for precision 10^{-30}, one thousand times smaller circuits, which is an expected difference between algorithms with scaling $O(\log^{3.97}(1/\varepsilon))$ and $O(\log(1/\varepsilon))$.

B. Performance evaluation

Our experiments confirm that the algorithm described in the paper has a probabilistic polynomial runtime. In addition, constants and the power of the polynomial are such that our algorithm is practically useful. In Figure 18c we show the runtime of different parts of our algorithm. The approximation part corresponds to the runtime of the algorithm approximating unitaries $R_z(\phi)$ by exact unitaries (Figure 12), excluding time needed to solve the norm equation; the synthesis part corresponds to the runtime of the exact synthesis algorithm that produces an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit; the resynthesis part corresponds to the runtime of peephole optimization performed on $\langle \sigma_1, \sigma_2 \rangle$-circuits obtained from $\langle \mathcal{F}, \mathcal{T} \rangle$-circuits using identities from Section III.

We separately show the runtime of the relative norm equation solver because for our implementation we used a generic solver which is a part of the PARI/GP library (function *rnfisnorm* [23]). Since the library documentation does not describe the function performance in detail, we performed an evaluation to confirm that it is polynomial time on "easy" instances of the problem. We also rely on another PARI/GP function *ispseudoprime* to perform a primality test. It is a combination of several probabilistic polynomial time primality tests [23]. The runtime of the primality test is included in the approximation part of the figure.

Our claim about the approximation algorithm runtime depends on Conjecture 20; Figure 18d shows the number of trials performed in the main loop of the approximation algorithm before finding an easy instance. The scaling of the average number of trials shown in the figure supports the conjecture. To perform peephole optimization [12] we used the database of optimal $\langle \sigma_1, \sigma_2 \rangle$-circuits with up to 19 gates which has size 85.7 MB.

High-precision integer and floating-point data types are necessary to implement our algorithm. We use *cpp_int* and *cpp_dec_float* from *boost::multiprecision* li-

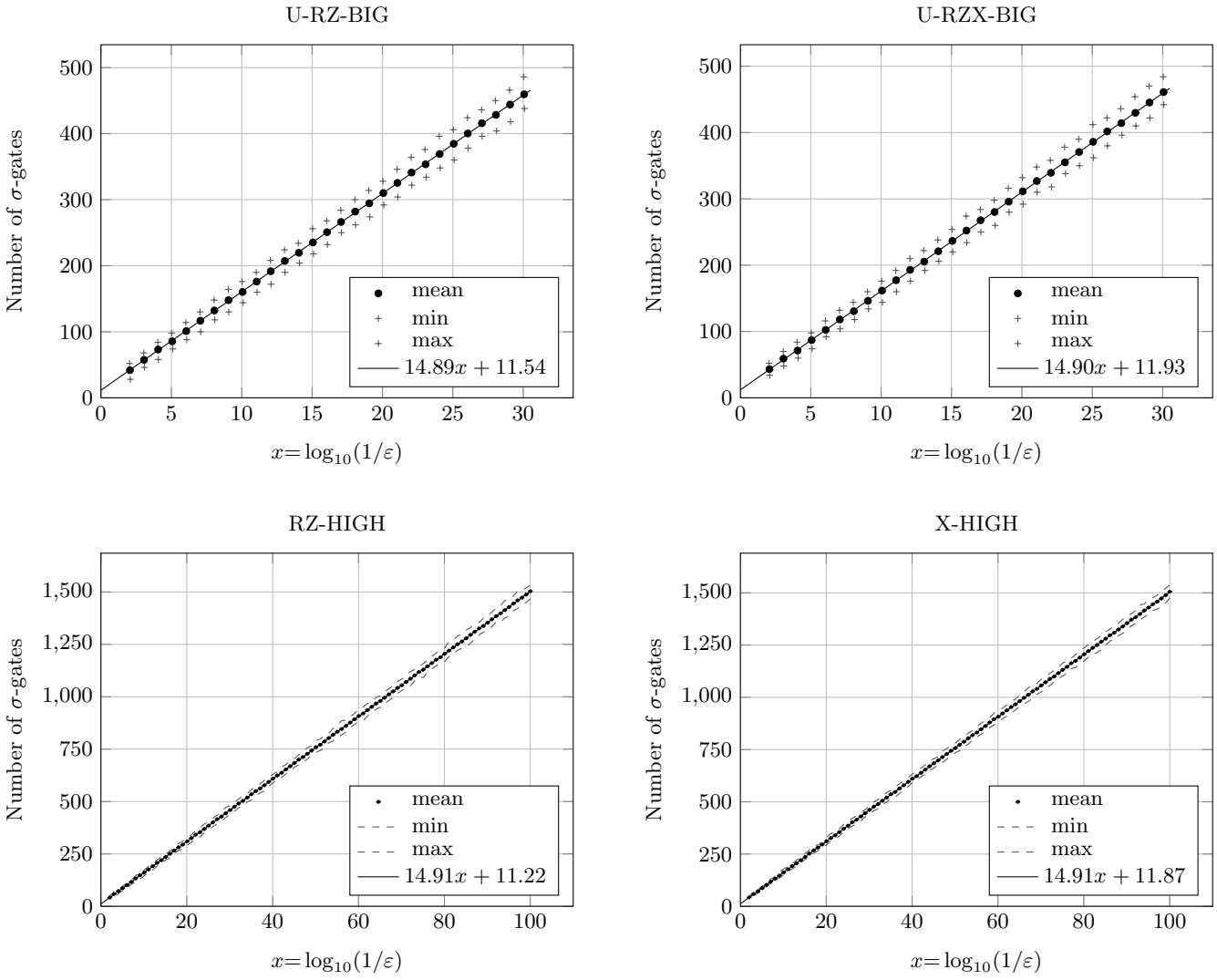

Figure 17: Number of σ gates needed to achieve the quality of approximation ε using the Number Theoretic Algorithm on different sets of inputs (see Table I). Includes approximation of X gate and R_z rotations used in the Quantum Fourier Transform.

brary. The number of bits used by these types can be specified at compile time. This allows us to avoid dynamic memory allocation when performing arithmetic operations; much faster stack memory is used instead. For this reason, runtime scaling (Figure 18c) of our code is a function of the number of arithmetic operations, and not of the bit size of numbers used in the algorithm.

In Figure 19a we show how the runtime of our algorithm changes when using different arithmetic types on the same set of inputs. The first pair of types — 512 bit integers and 200 decimal digits floating-point numbers — is sufficient for precision up to 10^{-35}, the second pair — 1024 bits and 400 decimal digits — for precision up to 10^{-70}. Figure 18c shows runtime scaling for different parts of our algorithm in more detail when using the second pair of types. This shows that our algorithm is practical and can readily be used as a subroutine when compiling quantum algorithms with large numbers of different single-qubit operations.

VII. CONCLUSIONS AND FUTURE WORK

We have considered the problem of optimal representation of single-qubit unitaries as braid patterns in the Fibonacci anyon basis, which is a promising non-Abelian anyon framework for topological computing. We have developed a set of principled solutions that enables the compilation of single-qubit unitaries into circuits of depth $O(\log(1/\varepsilon))$ for an arbitrary target precision ε; the compiled circuits do not require ancillary qubits or pre-compiled resource states and are asyptotically depth-optimal.

The compiler runs on a classical computer in running

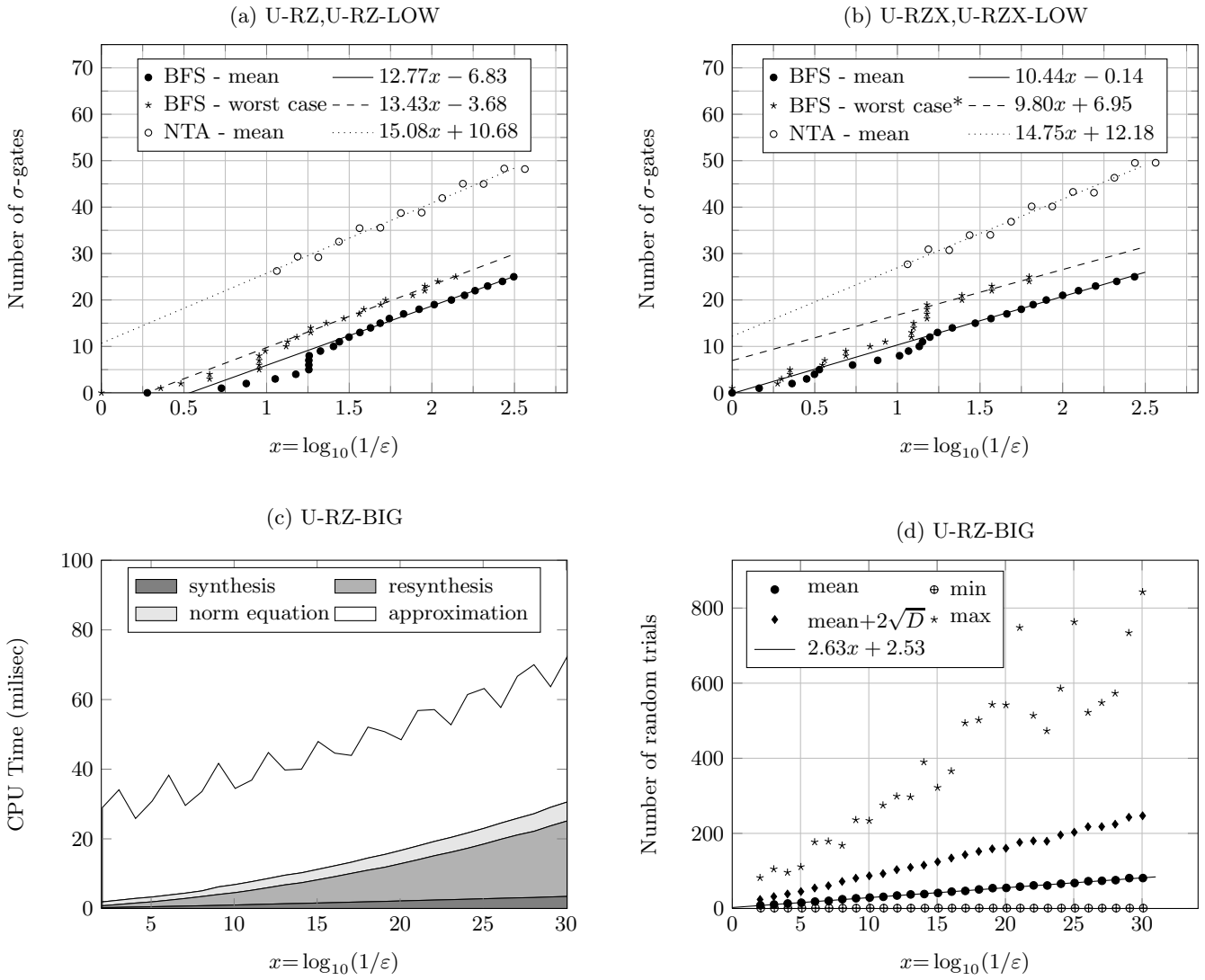

Figure 18: (a),(b) comparison of the number of σ gates needed to achieve the quality of approximation ε using the Number Theoretic Algorithm (NTA) and using Brute Force Search (BFS). Input sets U-RZ-LOW and U-RZX-LOW were used for the NTA and U-RZ, U-RZX for BFS; (c) average runtime of parts of the NTA, using U-RZ-BIG input set; (d) number of trials performed in the main loop of the NTA before an "easy" instance was found, using U-RZ-BIG input set. See Table I for input set descriptions.

time that is probabilistically polynomial in the bit size of ε. In practice, the runtime appears to scale slower than $\log^2(1/\varepsilon)$ when ε tends to zero. Compilation of an axial rotation to precision 10^{-30} takes less than 80 milliseconds on average on a regular classical desktop computer. Consequently, our compiler improves on the most recent state-of-the-art solutions (c.f., [10]) in both an asymptotic and practical sense.

The availability of an asymptotically optimal compiler for single-qubit unitaries over the Fibonacci anyon basis lays a foundation for solving more challenging problems in topological quantum compilation, such as the problem of finding asymptotically optimal representations of multi-qubit unitary operations by circuits over that basis. The fact that the multi-qubit Clifford group is not native to the Fibonacci anyon framework adds complexity to the challenge. It implies that either high-quality approximations of two-qubit Clifford gates must be designed or an entirely different set of entanglement gates must be considered, necessitating a translator between representations. Thus, our future work will be multi-qubit compilation; we hope that the number theoretical methods described in this paper can be leveraged in the multi-qubit context.

Another direction is compilation of approximation circuits into "weave" representations [9, 10]. A weave is a restricted braid where only one quasiparticle is allowed to move; a weave circuit is a circuit composed entirely of weaves. It has been shown [9, 10] that any braid pattern circuit can be approximated by a weave circuit. Achiev-

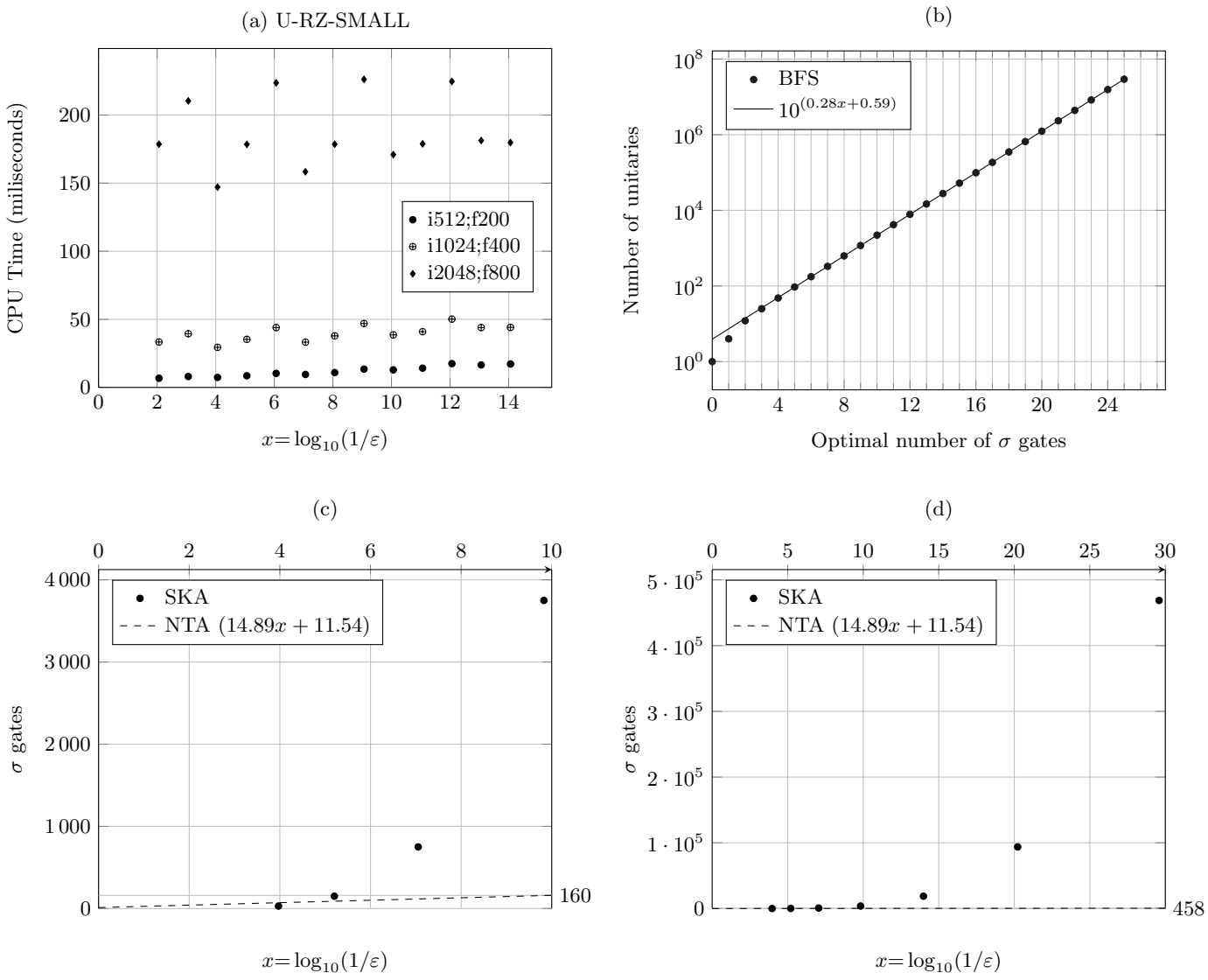

Figure 19: (a) runtime of the Number Theoretic Algorithm when using different arithmetic data types. Type i**N** corresponds to a signed, unchecked boost::multiprecision::cpp_int with MinDigits and MaxDigits set to **N**; type f**N** corresponds to boost::multiprecision::cpp_dec_float with Digits10 set to **N**; (b) number of distinct (up to a global phase) unitaries, such that their optimal implementation requires given number of σ gates. (c),(d) comparison of the estimated size of circuits produced by the Solovay-Kitaev algorithm (SKA) and the Number Theoretic Algorithm (NTA).

ing an end-to-end compilation into weaves in probabilistically polynomial runtime with minimal overhead is an important direction of our future work.

ACKNOWLEDGMENTS

We wish to thank Andreas Blass, Yuri Gurevich, Matthew Hastings, Martin Roeteller, Jon Yard and Dave Wecker for useful discussions. VK wishes to thank Dr. Cameron L. Stewart for helpful discussions regarding continued fractions.

[1] A. Kitaev, Ann.Phys. (N.Y.) **303**, 2 (2003).

[2] S. Sarma, M. Freedman, and C. Nayak, Phys.Rev **94**,

166802 (2005).
[3] C. Nayak, S. Simon, A. Stern, M. Freedman, and S. Sarma, Rev.Mod.Phys **80**, 1083 (2008).
[4] S.-S. Chern and J. Simmons, Annals of Mathematics **99(1)**, 48 (1974).
[5] E. Witten, Commun. Math. Phys. **117**, 353 (1988).
[6] E. Witten, Commun. Math. Phys. **121(3)**, 351 (1989).
[7] M. Freedman, M. Larsen, and Z. Wang, Comm. Math. Phys. **227(3)**, 605 (2002).
[8] M. Freedman, M. Larsen, and Z. Wang, Comm. Math. Phys. **228**, 177 (2002).
[9] S. Simon, N. Bonesteel, M. Freedman, N. Petrovic, and L. Hormozi, Phys. Rev. Lett. **96**, 070503 (2006).
[10] L. Hormozi, G. Zikos, N. Bonesteel, and S. Simon, Phys. Rev. B **75**, 165310 (2007).
[11] C. Dawson and M. Nielsen, Quantum Information and Computation **6**, 81 (2006).
[12] A. K. Prasad, V. V. Shende, I. L. Markov, J. P. Hayes, and K. N. Patel, ACM Journal on Emerging Technologies in Computing Systems **2**, 277 (2006).
[13] P. Selinger, (2012), arXiv:1212.6253.
[14] V. Kliuchnikov, D. Maslov, and M. Mosca, (2012), arXiv:1206.5236.
[15] V. Kliuchnikov, D. Maslov, and M. Mosca, (2012), arXiv:1212.6964.
[16] A. Bocharov, Y. Gurevich, and K. Svore, Phys. Rev. A **88**, 012313 (2013).
[17] S. Trebst, M. Troyer, Z. Wang, and A. Ludwig, Prog. Theor. Phys. Supp. **176**, 384 (2008).
[18] J. Preskill, *Lecture Notes for Physics 219:Quantum Computation* (2004).
[19] J. H. Lenstra, Journal London Math. Soc. **10**, 457 (1975).
[20] N. Jacobson, *Basic Algebra I* (Dover, 2009).
[21] H. Cohen, *Advanced Topics in Computational Algebraic Number Theory* (Springer, 1999).
[22] D. Simon, Mathematics of Computation **Vol. 71, No. 239**, 1287 (2002).
[23] T. P. Group, *User's Guide to PARI/GP* (Institut de Mathematiques de Bordeaux, 2011).
[24] D. Shanks, Proceedings of the Second Manitoba Conference on Numerical Mathematics , 51 (1973).
[25] H. Cohen, *A Course in Computational Algebraic Number Theory* (Springer, 1996).
[26] E. Bach, Mathematics of Computation **Vol. 55, No. 191**, 355 (1990).
[27] I. Damgard and G. Frandsen, J. Symbolic Computation (2005).
[28] D. Wikstrom, Automata, Languages and Programming, LNCS **Vol. 3580**, 1189 (2005).
[29] A. W. Harrow, B. Recht, and I. L. Chuang, J. Math. Phys. **43** (2002).
[30] H. Cohen, *Number Theory, Volume I: Tools and Diophantine Equations* (Springer, 2000).
[31] L. Washington, *Introduction to Cyclotomic Fields* (Springer,New-York, 1997).
[32] J. Neukirch, *Algebraic Number Theory* (Springer,Berlin, 1999).

Appendix A: Background on Number Field Extensions

We present the key mathematical concepts needed to understand the number theory involved in our algorithms. A systematic discourse of underlying mathematics can be found in [25, 30].

In this section, a "field" means by default a *number field* which is, by definition, a finite extension of the field of rational numbers $\mathbb{Q}$.

The particular fields we work with in this paper are the cyclotomic field $\mathbb{Q}(\omega)$, where $\omega = e^{\pi\, i/5}$ is the tenth primitive root of unity, and its maximum real subfield $\mathbb{Q}(\tau)$, where $\tau = (\sqrt{5}-1)/2$ is the inverse of the golden ratio. It is worth noting that there is some freedom in how we represent a field. For example, $\mathbb{Q}(\tau)$ can be alternatively generated by the golden ratio $\phi = (\sqrt{5}+1)/2$ or by $\sqrt{5}$. In the same vein, $\mathbb{Q}(\omega)$ can be alternatively generated by $\theta = \omega + \omega^4$. In these terms it is easier to see that $\mathbb{Q}(\omega)$ is a quadratic extension of $\mathbb{Q}(\tau)$; indeed, $\theta^2 = \tau - 2$. Similarly, $\mathbb{Q}(\tau)$ is a quadratic extension of $\mathbb{Q}$; indeed, τ is a root of the monic polynomial $T^2 + T - 1 = 0$.

1. Galois extension and relative norm

Let L/K be an algebraic field extension of order n.

Definition 23. *A K-linear map $f : L \to L$ is called a **K-automorphism** if it is bijective and preserves field multiplication, i.e., $\forall a, b \in L, f(a\,b) = f(a)\,f(b)$.*

Clearly a composition of two K-automorphisms is a K-automorphism and the inverse of a K-automorphism ia a K-automorphism.

Definition 24. *Field extension L/K of order n is called normal or **Galois** if there exists exactly n K-automorphisms of L. The set of all K-automorphisms of L forms a group under composition that is called the **Galois group of** L/K, denoted by $Gal(L/K)$.*

We note the following:

Proposition 25. *If L/K is a Galois extension then K coincides with the set of fixed points of the group $Gal(L/K)$.*

Even though the notion of *relative norm* is usually defined in a more general setting, we focus on the particular definition best suited for the Galois extensions.

Definition 26. *Given a Galois extension L/K the **relative norm** maps $N_{L/K} : L \to K$ and is defined as follows*

$$N_{L/K}(\alpha) = \prod_{\sigma \in Gal(L/K)} \sigma(\alpha). \tag{A1}$$

Correctness of this definition follows from proposition 25: since $N_{L/K}(\alpha)$ is a fixed point of the Galois group, it belongs to K. It is also easy to see that $\forall \alpha, \beta \in L, N_{L/K}(\alpha\,\beta) = N_{L/K}(\alpha)\,N_{L/K}(\beta)$.

Elements $\{\sigma(\alpha) | \sigma \in Gal(L/K)\}$ are commonly called (Galois) conjugates of the element $\alpha \in L$. Thus, the relative norm of an $\alpha \in L$ is the product of all its Galois conjugates. For an element $\beta \in K \subset L$, all the Galois conjugates are the same and coincide with β. Therefore $N_{L/K}(\beta) = \beta^n$, where n is the order of the extension (and thus the size of the Galois group), and also for $\beta \in K \subset L$ and $\alpha \in L$, $N_{L/K}(\beta\,\alpha) = \beta^n\,\alpha$.

2. Galois group and relative norm in $\mathbb{Z}[\omega]$

Let ω be the tenth root of unity as defined in Sectoin II). Consider the field extension $\mathbb{Q}(\omega)/\mathbb{Q}$, which is, by definition, the smallest in $\mathbb{C}$ containing both the rational numbers $\mathbb{Q}$ and the ω.

Considering the fifth root of unity $\zeta_5 = e^{2\,i\,\pi/5} = \omega^2$, it is well known that that $\mathbb{Q}(\omega)$ coincides with the $\mathbb{Q}(\zeta_5)$, which is recognized as one of the simpler *cyclotomic fields* (see [31]). The minimal monic polynomial of ζ_5 over $\mathbb{Q}$ is $X^4 - X^3 + X^2 - X + 1$, which can be easily verified by factoring $X^{10} - 1$ over $\mathbb{Z}$. This means that $\mathbb{Q}(\omega)$ is fourth-degree extension of $\mathbb{Q}$. When viewed as a vector space over $\mathbb{Q}$, the $\mathbb{Q}(\omega)$ is a vector space with basis $\{1, \omega, \omega^2, \omega^3\}$ and ω is an algebraic integer in $\mathbb{Q}(\omega)$. It is known (see [30]) that the ring of algebraic integers of $\mathbb{Q}(\omega)$ coincides with $\mathbb{Z}[\omega]$ and that the latter also has $\{1, \omega, \omega^2, \omega^3\}$ as its integral basis.

Let us view complex conjugation as a field automorphism:

$$(\cdot)^* : \mathbb{Q}(\omega) \to \mathbb{Q}(\omega). \tag{A2}$$

By direct computation

$$\omega^* = \omega^{-1} = 1 - \omega + \omega^2 - \omega^3 \tag{A3}$$

is linear with integer coefficients. Therefore its restriction to $\mathbb{Z}[\omega]$ is a ring automorphism:

$$(\cdot)^* : \mathbb{Z}[\omega] \to \mathbb{Z}[\omega]\,. \tag{A4}$$

Consider the subring $\mathbb{Z}[\tau] \subset \mathbb{Z}[\omega]$ which is the minimal subring of $\mathbb{Z}[\omega]$ containing $\tau = \omega^2 - \omega^3$. Since $\mathbb{Z}[\tau]$ is populated by real-valued elements, it remains fixed under complex conjugation. Consider $\mathbb{Q}(\tau) \subset \mathbb{Q}(\omega)$, the field of fractions of the ring $\mathbb{Z}[\tau]$. (Conversely, $\mathbb{Z}[\tau]$ is the ring of integers of the field $\mathbb{Q}(\tau)$.) As per [31]:

(1) $\mathbb{Q}(\omega)/\mathbb{Q}(\tau)$ is a Galois extension with the two-element Galois group consisting of identity and the complex conjugation.

(2) Hence $\mathbb{Q}(\tau) = \mathbb{Q}(\omega) \cap \mathbb{R}$ is the maximum real subfield of $\mathbb{Q}(\omega)$.

Similarly, $\mathbb{Z}[\tau] = \mathbb{Z}[\omega] \cap \mathbb{R}$. Since the minimal polynomial of τ is $X^2 + X - 1$, $\mathbb{Q}(\tau)/\mathbb{Q}$ is a quadratic extension. The ring $\mathbb{Z}[\tau]$ and its fraction field $\mathbb{Q}(\tau)$ play an extremely important role in most of our constructions.

Consider the ladder of extensions $\mathbb{Q}(\omega) \supset \mathbb{Q}(\tau) \supset \mathbb{Q}$. The ring automorphism already introduced in (5) extends to field automorphism in a natural way:

$$(\cdot)^\bullet : \mathbb{Q}(\omega) \to \mathbb{Q}(\omega);\ (\omega)^\bullet = \omega^3. \tag{A5}$$

The Galois group of the fourth-order extension $\mathbb{Q}(\omega)/\mathbb{Q}$ happens to be the cyclic group $\mathbb{Z}_4$ generated by the $()^\bullet$, in particular the complex conjugation is $()^{\bullet\bullet}$. By direct computation,

$$\tau^\bullet = -(\tau + 1), \tag{A6}$$

therefore the $()^\bullet$ automorphism has correctly defined restrictions to both $\mathbb{Q}(\tau)$ and $\mathbb{Z}[\tau]$. Since both are quadratic over rationals (resp., over rational integers) the restriction must be an order two automorphism, which is also seen directly since complex conjugation is the identity on $\mathbb{Q}(\tau)$.

We now define the norm maps for all the above field extensions:

$$N_\tau : \mathbb{Q}(\tau) \to \mathbb{Q};\ N_\tau(\xi) = \xi\,\xi^\bullet \tag{A7}$$

$$N_i : \mathbb{Q}(\omega) \to \mathbb{Q}(\tau);\ N_i(\eta) = \eta\,\eta^* \tag{A8}$$

$$N : \mathbb{Q}(\omega) \to \mathbb{Q};\ N(\eta) = N_\tau N_i(\eta) \tag{A9}$$

The properties of the complex conjugation and the $()^\bullet$ automorphism imply that all the norm maps have correctly defined restrictions to the respective rings of integers $\mathbb{Z}[\omega]$ and $\mathbb{Z}[\tau]$.

We are now ready to introduce the *Gauss complexity measure* on the ring $\mathbb{Z}[\omega]$:

$$G : \mathbb{Z}(\omega) \to \mathbb{Z};\ G(\eta) = N_i(\eta) + N_i(\eta)^\bullet \tag{A10}$$

(see [19] for the intuition behind this measure). The correctness of this definition follows from the fact that $G(\eta)^\bullet = G(\eta)$ and therefore $G(\eta)$ is rational, but also since both N_i and $(\cdot)^\bullet$ are integral in the integral bases, it has to be a rational integer.

Appendix B: Background on Relative Norm Equation

We introduce just enough algebraic number theory to handle Thm 10 in a principled way. The theorem offers a solvability condition (2) that is best derived from the theory of cyclotomic fields as presented in [31]. It also offers a specific form for a solution (statement (b)) that can be derived from prime ideal decomposition algorithm, best described in [32].

1. Prime ideals in a Galois extension

Below the term "ring" will stand for a commutative ring with unity. We start with common definitions (c.f., [30]).

Definition 27. *(1) An additive subgroup I of a ring R is called an **ideal** if it is closed under multiplication by any element of $r \in R$, i.e., $r\,I = \{r\,a | a \in I\} \subset I$.*

*(2) The **sum** $I + J$ of two ideals I and J of a ring R consists of pairwise sums $a + b, a \in I, b \in J$.*

*(3) The **product** $I\,J$ of two ideals I and J of a ring R consists of finite sums of pairwise products $a\,b, a \in I, b \in J$.*

Both the sum and the product of two ideals of a ring is an ideal of that ring. It is also easy to see that (1) summation of ideals is associative and commutative, (2) multiplication of ideals is associative, commutative and distributive with respect to summation of ideals, (3) that the ideal generated by zero is the neutral element with respect to the addition of the ideals and (4) the entire ring R is an ideal that is the neutral element with respect to multiplication of its ideals.

We call an ideal $\mathcal{I} \subset R$ a *proper* ideal if it does not coincide with the entire ring R.

Definition 28. *Given an element of a ring R, $g \in R$, the ideal $gR = \{g\,r | r \in R\}$ is the **principal** ideal generated by the element g.*

Definition 29. *An ideal I of a ring R is called a **prime** ideal if it cannot be represented as a product of two or more proper ideals of the ring R.*

Proposition 30. *A principal ideal gR of a ring R is prime if and only if g is a prime element of the ring R.*

Consider a Galois field extension L/K and let $\mathcal{O}_L$and $\mathcal{O}_K$ be the corresponding rings of integers.

Definition 31. *A prime ideal $\mathcal{P} \subset \mathcal{O}_K$ is*

*(1) **inert** in $\mathcal{O}_L$ if $\mathcal{P}\mathcal{O}_L$ is a prime ideal in $\mathcal{O}_L$,*

*(2) **fully ramified** in $\mathcal{O}_L$ if $\mathcal{P}\mathcal{O}_L = (\mathcal{Q})^d$ where $\mathcal{Q}$ is a prime ideal in $\mathcal{O}_L$,*

*(3) **fully split** in $\mathcal{O}_L$ if $\mathcal{P}\mathcal{O}_L = \prod_{j=1}^{d} \mathcal{Q}_j$ where $\mathcal{Q}_j, j = 1, ..., d$ are pairwise distinct prime ideals in $\mathcal{O}_L$.*

It follows from field extension theory (c.f., [30]) that the exponent d in clauses (2),(3) of Definition 31 is equal to the order of the Galois group $Gal(L/K)$.

2. Split primes in the cyclotomic field $\mathbb{Q}(\omega)$

We apply these concepts to the number fields addressed in this paper. We have a ladder of field extensions $\mathbb{Q}(\omega)/\mathbb{Q}(\tau)$ and $\mathbb{Q}(\tau)/\mathbb{Q}$ and corresponding extensions of the rings of integers $\mathbb{Z}[\omega]/\mathbb{Z}[\tau]$, $\mathbb{Z}[\tau]/\mathbb{Z}$. All the listed extensions are quadratic extensions and the *absolute* extensions $\mathbb{Q}(\omega)/\mathbb{Q}$ and $\mathbb{Z}[\omega]/\mathbb{Z}$ are order 4. All the Galois groups of all extensions above are cyclic.

Recall that $\mathbb{Q}(\omega)$ coincides with the *cyclotomic* field $\mathbb{Q}(\zeta_5)$ where ζ_5 is the fifth primitive root of unity. It is easy to establish that an inert prime of the extension $\mathbb{Z}[\tau]/\mathbb{Z}$ is a rational integer prime p such that $p = \pm 2 \bmod 5$. It is also easy to prove that given a rational integer prime p such that $p = \pm 1 \bmod 5$, it is going to be fully split in the extension $\mathbb{Z}[\tau]/\mathbb{Z}$. That is, there exists a $\xi \in \mathbb{Z}[\tau]$ such that $N_\tau(\xi) = N_\tau(\xi^\bullet) = p$.

Proposition 32. *In the above context, given $\xi > 0$ and $p = N_\tau(\xi)$ is prime, the relative norm equation $N_i(x) = \xi$ has a solution if and only if p is fully split in the absolute extension $\mathbb{Z}[\omega]/\mathbb{Z}$.*

Proof. Indeed, if $N_i(x_1) = \xi$ and x_2, x_3, x_4 are $Gal(\mathbb{Q}(\omega)/\mathbb{Q})$ conjugates of x_1, then $N(x_1) = x_1\,x_2\,x_3\,x_4 = p$, where $N = N_\tau\,N_i$ is the absolute norm. Therefore p is fully split.

The converse is also true under the standing condition that $\xi > 0$. Suppose $p = x_1\,x_2\,x_3\,x_4$ is fully split. By relabeling we can ensure that $x_2 = x_1^\bullet$, $x_3 = x_2^\bullet = x_1^*$, $x_4 = x_3^\bullet = x_2^*$. Let $\eta = N_i(x_1) = x_1\,x_3$, then $x_2\,x_4 = N_i(x_2) = N_i(x_1)^\bullet = \eta^\bullet$. Hence $p = \eta\,\eta^\bullet = N_\tau(\eta)$. The pair of τ-conjugate solutions of $N_\tau(\eta) = p$ is unique up to units of a certain form, specifically $N_\tau(\xi) = N_\tau(\eta)$ means that either $\xi = \pm\tau^{2k}\,\eta$ or $\xi = \pm\tau^{2k}\eta^\bullet$. However, both η and $\eta^\bullet$ are square norms of complex entities and thus both are positive. Given $\xi > 0$, $\xi = +\tau^{2k}\,\eta$ or $\xi = +\tau^{2k}\eta^\bullet$. In the first case, $N_i(\tau^k\,x_1) = \xi$ and in the second case $N_i(\tau^k\,x_2) = \xi$. □

3. Proof of Statement (2) of Theorem 10

We prove the statement by invoking Theorem 2.13 of [31].

In the case of the cyclotomic field $\mathbb{Q}(\zeta_5)$, if $\xi \in \mathbb{Z}[\tau]$ happens to be an inert integer prime of $\mathbb{Z}[\tau]/\mathbb{Z}$ then it is also inert in $\mathbb{Z}[\zeta_5]/\mathbb{Z}[\tau]$. Indeed, in this case ξ is a rational integer prime and $\xi = \pm 2 \bmod 5$. The smallest power f such that $\xi^f = 1 \bmod 5$ is 4 and coincides with the order of the $\mathbb{Z}[\zeta_5]/\mathbb{Z}$ extension. Thus ξ does not split.

Otherwise, for the prime ξ either $p = N_\tau(\xi) = 5$ or $p = N_\tau(\xi) = \pm 1 \bmod 5$. The easy case of $p = 5$ is covered by Observation 9. In the remaining case p must be fully split in $\mathbb{Z}[\zeta_5]/\mathbb{Z}$ as per Proposition 32. As per the cited Theorem 2.13, this is equivalent to $p = 1 \bmod 5$, which is precisely what is claimed in Statement (2) of Thm 10.

4. Prime decomposition of ideal in Galois extension

We now prove Statement (b) of Thm 10. This requires a textbook fact about *fully split* ideals and an algorithm that deals with such ideals.

Let L/K be a Galois field extension.

Theorem 33. *(c.f., [32]) If a prime ideal $\mathcal{P} \subset \mathcal{O}_K$ is fully split in $\mathcal{O}_L$then $\mathcal{P}$ is uniquely (up to relabeling) represented as a product of distinct ideals $\mathcal{P}\mathcal{O}_L = \prod_{j=1}^{d} \mathcal{Q}_j$, where $\mathcal{Q}_j, j = 1, ..., d$ are prime in $\mathcal{O}_L$. The Galois group $Gal(L/K)$ acts faithfully and transitively in this set of prime ideals. In particular, d is the order of the Galois group.*

Note that, in case $\mathbb{Q}[\omega]/\mathbb{Q}[\tau]$, $d=2$ and the two ideals occurring in the prime decomposition of a split prime ideal are complex conjugates of each other.

Definition 34. *$\theta \in L$ is a **primitive element** of the extension L/K if $L = K(\theta)$.*

In our case the primitive element of choice for the $\mathbb{Q}(\omega)/\mathbb{Q}(\tau)$ extension is $\theta = \omega+\omega^4 = -1+2\,\omega-\omega^2+\omega^3 = \sqrt{\tau-2}$.

Definition 35. *For a primitive element θ of the extension L/K, consider the conductor ideal $\mathcal{C}_\theta = \{y \in \mathcal{O}_L | y\,\mathcal{O}_L \subset \mathcal{O}_K[\theta]\}$. A prime ideal $\mathcal{P} \subset \mathcal{O}_K$ is **exceptional with respect to** θ when it is not relatively prime with the $\mathcal{C}_\theta$.*

For the above choice of primitive element θ in $\mathbb{Q}(\omega)/\mathbb{Q}(\tau)$, $\mathcal{C}_\theta = 2\mathbb{Z}[\omega]$, so the only exceptional prime ideal is the inert ideal (2). There are no exceptional split prime ideals in this case.

Algorithm 36. *(c.f., [32]) Let $\mathcal{P} \subset \mathcal{O}_K$ be a non-exceptional prime ideal and let $H(X) \in \mathcal{O}_K[X]$ be the monic minimal polynomial for the primitive element θ. Let $F = \mathcal{O}_K/\mathcal{P}$ be the corresponding residue field and let $h(X) \in F[X]$ be the reduction of the $H(X)$ modulo $\mathcal{P}$. Compute the irreducible factorization $h(X) = h_1(X)^{e_1}...h_r(X)^{e_r}$ in $F[X]$. Then*

$$\mathcal{P}\mathcal{O}_L = \mathcal{Q}_1^{e_1}...\mathcal{Q}_r^{e_r},$$

where $\mathcal{Q}_j = \mathcal{P}\mathcal{O}_L + h_j(\theta)\mathcal{O}_L, j = 1, ..., r$.

Corollary 37. *(Special Case.) In the context of Thm 33, let $\mathcal{P} \subset \mathcal{O}_K$ be a prime ideal that is split in $\mathcal{O}_L$ and non-exceptional with respect to a primitive element θ of L/K. Then*

(1) $F = \mathcal{O}_K/\mathcal{P}$ is a splitting field of the residual mimimal polynomial, i.e., $h(X) = (X-m_1)...(X-m_d)$ in $F[X]$;

(2) If additionally $\mathcal{O}_L$ is a principal ideal domain then we can effectively find such $s \in \mathcal{O}_L$ that $\mathcal{P}\mathcal{O}_L = N_{L/K}(s)\mathcal{O}_L$. That is, we can solve the relative norm equation in the group of ideals.

Indeed, (1) is a straightforward consequence of the algorithm. For (2), consider the ideal $\mathcal{Q}_1 = \mathcal{P}\mathcal{O}_L + (\theta - m_1)\mathcal{O}_L$. Since $\mathcal{O}_L$ is a principal ideal domain, $\mathcal{Q}_1 = s_1\mathcal{O}_L$ for some $s_1 \in \mathcal{O}_L$. Recall that the Galois group $Gal(L/K)$ acts faithfully and transitively on the set of ideals $\{\mathcal{Q}_j\}$, let $\sigma_j \in Gal(L/K)$ be the element mapping ideal $\mathcal{Q}_1$ onto the ideal $\mathcal{Q}_j, j = 2, .., d$ and let $s_j = \sigma_j(s_1), j = 2, ..., d$. Obviously for each j, $\mathcal{Q}_j = s_j\mathcal{O}_L$ and $\mathcal{P}\mathcal{O}_L = \prod_{j=1}^d \mathcal{Q}_j = (\prod_{j=1}^d s_j)\mathcal{O}_L = N_{L/K}(s_1)\mathcal{O}_L$.

As a concluding observation, let $\mathcal{P}\mathcal{O}_L$ be the principal ideal $\xi\mathcal{O}_L$ for some $\xi \in \mathcal{O}_K$. Then the above equality of principal ideals means that $\xi \sim N_{L/K}(s_1)$, i.e., $\xi = u\,N_{L/K}(s_1)$ for some unit u. Since $\xi, N_{L/K}(s_1)$ are in $\mathcal{O}_K$, so is the unit u.

5. Proof of Statement (b) of Theorem 10

For brevity, we leave out the easy case of $p = N_\tau(\xi) = 5$. The minimal polynomial for θ is $H(X) = X^2-(\tau-2)$, and its reduction $h(X) = X^2 - ((\tau-2) \bmod p)$. If conditions (1),(2) of the theorem hold, and in particular $p = 1 \bmod 5$, $h(X)$ is guaranteed to split in $\mathbb{Z}_p$ into $h(X) = (X-m)(X+m) \bmod p, m \in \mathbb{Z}$.

As per Algorithm 36, the ideal $\xi\mathbb{Z}[\omega]$ is the product of $\xi\mathbb{Z}[\omega]+(m-\theta)\mathbb{Z}[\omega]$ and $\xi\mathbb{Z}[\omega]+(m+\theta)\mathbb{Z}[\omega]$. As per the discussion around the Corollary 37, $\xi = u\,N_i(s)$, where u is a unit in $\mathbb{Z}[\tau]$ and s is (for example) a generator of the principal ideal $\xi\mathbb{Z}[\omega] + (m-\theta)\mathbb{Z}[\omega]$. We can take $s = GCD_{\mathbb{Z}[\omega]}(\xi, m-\theta)$ as such a generator. Note that the necessary conditions of solvability of (15), $\xi > 0, \xi^\bullet > 0$ in this specific context turn out to be sufficient. Indeed, they imply that $u > 0, u^\bullet > 0$ which makes u equal to τ^{2k} for some $k \in \mathbb{Z}$. Then $x = \tau^k\,s$ is the desired solution of (15). It follows that the other solution is $x^* = \tau^k\,s^*$.

6. Proof of Theorem 11

To prove the theorem, we first need the following:

Lemma 38. *Let*

$$\xi = \eta^2\xi_1...\xi_r, \eta, \xi_j \in \mathbb{Z}[\tau], r \in \mathbb{Z}, r \geq 0, j = 1, ..., r \quad \text{(B1)}$$

be a factorization of an $\xi \in \mathbb{Z}[\tau]$, where $\xi_1, ..., \xi_r$ are $\mathbb{Z}[\tau]$-primes such that $\xi_1\mathbb{Z}[\tau], ..., \xi_r\mathbb{Z}[\tau]$ are pairwise distinct prime ideals. If $\xi > 0, \xi^\bullet > 0$ then there exists an equivalent factorization where $\xi_j > 0, \xi_j^\bullet > 0$ for each $j = 1, ..., r$

Proof. For any choice of $\xi_1, ..., \xi_r$, there is an even number of these that are negative. Replacing each negative ξ_j by $-\xi_j$ we get an equivalent factorization of ξ. Assuming further that $\xi_1, ..., \xi_r$ are all positive, we note that there is an even number of these that have negative adjoint. Now split the set $\{\xi_j | \xi_j^\bullet < 0\}$ arbitrarily into two halves and replace each ξ_j in the first half by $\tau\,\xi_j$ (ensuring $(\tau\,\xi_j)^\bullet > 0$) then replace each ξ_j in the second half by $\tau^{-1}\xi_j$(ensuring $(\tau^{-1}\xi_j)^\bullet > 0$). Then this set of replacements leads to an equivalent factorization of ξ. □

We are finally ready to prove Thm 11. Given a factorization (B1) and assuming as per the lemma that $\xi_j > 0, \xi_j^\bullet > 0$ for each $j = 1, ..., r$, the theorem can be restated to claim that the equation $N_i(x) = \xi$ is solvable if and only if all the equations $N_i(y) = \xi_j$ are individually solvable.

Proof. If the factor equations are individually solvable, then the $N_i(x) = \xi$ is obviously solvable.

Conversely, suppose (15) is solvable for the square-free ξ, i.e., $\xi = \xi_1...\xi_r = N_i(x), x \in \mathbb{Z}[\omega]$ and additionally $\xi_j > 0, \xi_j^\bullet > 0$ as explained above. We would like to show that each of the equations $N_i(x_j) = \xi_j$ is individually solvable, $j = 1, ..., r$. Consider the ideal $\xi_j\mathbb{Z}[\omega] + x\,\mathbb{Z}[\omega]$

and its principal ideal representation $s_j\,\mathbb{Z}\,[\omega]$. The complex conjugation of the ideal is $\xi_j\mathbb{Z}\,[\omega] + x^*\,\mathbb{Z}\,[\omega]$ and is equal to $s_j^*\,\mathbb{Z}\,[\omega]$. Since $N_i(s_j) = s_j s_j^*$ generates the product of the two ideals, it divides ξ_j^2. Since both $N_i(s_j)$ and ξ_j^2 are in $\mathbb{Z}\,[\tau]$, $N_i(s_j)$ divides ξ_j^2 in $\mathbb{Z}\,[\tau]$. Since ξ_j is prime, $N_i(s_j)$ is associate to either ξ_j^2 or to ξ_j. The former is impossible, since $N_i(s_j)$ also divides $x\,x^* = \xi$ and ξ is square free. Thus $u\,N_i(s_j) = \xi_j$, where u is a unit. By already familiar argument, $u > 0, u^\bullet > 0$ and thus $u = t^2$ is a perfect square. Therefore $N_i(t\,s_j) = \xi_j$. □