Quantum Information and Computation, Vol. 6, No. 1 (2006) 081–095
© Rinton Press

THE SOLOVAY-KITAEV ALGORITHM

CHRISTOPHER M. DAWSON
School of Physical Sciences, The University of Queensland,
Brisbane, Queensland 4072, Australia

MICHAEL A. NIELSEN
School of Physical Sciences, The University of Queensland,
Brisbane, Queensland 4072, Australia

Received May 6, 2005
Revised August 23, 2005

This pedagogical review presents the proof of the Solovay-Kitaev theorem in the form of an efficient classical algorithm for compiling an arbitrary single-qubit gate into a sequence of gates from a fixed and finite set. The algorithm can be used, for example, to compile Shor's algorithm, which uses rotations of $\pi/2^k$, into an efficient fault-tolerant form using only Hadamard, controlled-NOT, and $\pi/8$ gates. The algorithm runs in $O(\log^{2.71}(1/\epsilon))$ time, and produces as output a sequence of $O(\log^{3.97}(1/\epsilon))$ quantum gates which is guaranteed to approximate the desired quantum gate to an accuracy within $\epsilon > 0$. We also explain how the algorithm can be generalized to apply to multi-qubit gates and to gates from $SU(d)$.

Keywords: Solovay-Kitaev algorithm, universality, fault-tolerance
Communicated by: R Cleve & M Mosca

1 Introduction

"I have been impressed by numerous instances of mathematical theories that are really about particular algorithms; these theories are typically formulated in mathematical terms that are much more cumbersome and less natural than the equivalent formulation today's computer scientists would use." — Donald E. Knuth [17]

The Solovay-Kitaev (SK) theorem is one of the most important fundamental results in the theory of quantum computation. In its simplest form the SK theorem shows that, roughly speaking, if a set of single-qubit quantum gates generates a dense subset of $SU(2)$, then that set is guaranteed to fill $SU(2)$ *quickly*, i.e., it is possible to obtain good approximations to any desired gate using surprisingly short sequences of gates from the given generating set.

The SK theorem is important if one wishes to apply a wide variety of different single-qubit gates during a quantum algorithm, but is restricted to use gates from a more limited repertoire. Such a situation arises naturally in the context of fault-tolerant quantum computation, where direct fault-tolerant constructions are typically available only for a limited number of gates (e.g., the Clifford group gates and $\pi/8$ gate), but one may wish to implement a wider variety of gates, such as the $\pi/2^k$ rotations occurring in Shor's algorithm [23, 24]. In this situation one must use the limited set of fault-tolerant gates to build up accurate fault-tolerant approximations to all the gates used in the algorithm, preferably in the most efficient manner possible. The SK theorem tells us that such efficient constructions are possible.

The purpose of the present paper is to present the proof of the SK theorem in the form of a concrete *algorithm* which can be used to approximate a desired unitary, U, using a given set of quantum gates. We believe that this algorithmic perspective is useful in its own right, and, as the quote by Knuth suggests, also makes the ideas behind the SK theorem considerably easier to understand than in previous presentations. However, we stress that the paper is a review paper, as all the essential ideas appear in previous work; a detailed history of the SK theorem and algorithm is given in Section 6.

The structure of the paper is as follows. Section 2 introduces the fundamental definitions needed to understand the SK theorem and algorithm, and provides a formal statement of the SK theorem. Section 3 explains the SK algorithm for qubits, in the form of nine easily-understood lines of pseudocode. This section concentrates on developing a broad understanding of how the SK algorithm works. The few remaining details of the algorithm which need verification are explained in Section 4. Section 5 extends the algorithm so it applies also to *qudits*, i.e., quantum systems with d-dimensional state spaces. As a special case we obtain a version of the SK algorithm applicable to multiple qubits, simply by setting $d = 2^n$. Section 6 discusses the relationship of the results we have presented to prior work on the SK theorem. This placement of the discussion of prior work is perhaps somewhat unusual, and deserves comment: we chose to defer the discussion until late in the paper in order to make the comparisons between the present paper and prior work as concrete and transparent as possible. Section 7 concludes.

2 Fundamentals

In this section we introduce the fundamental definitions necessary to understand the SK theorem and algorithm, and give a precise statement of the SK theorem.

The basic goal of the SK algorithm is to take an arbitrary quantum gate U and find a good approximation to it using a sequence of gates $g_1, \ldots, g_m$ drawn from some finite set $\mathcal{G}$. In analogy with classical computation, where an algorithm written in a programming language such as C or Perl must be compiled into a sequence of instructions in the native machine language, we will call the set $\mathcal{G}$ of quantum gates an *instruction set*, and the process of finding good approximations *compilation*. Note that while this nomenclature, introduced in [10] (c.f. [11]), is inspired by classical computer science, the analogies between the classical and quantum concepts are obviously somewhat imprecise, and should not be taken too seriously.

Let's define these concepts more precisely, beginning with instruction sets:

Definition 1 *An* ***instruction set*** $\mathcal{G}$ *for a d-dimensional qudit is a* finite *set of quantum gates satisfying:*

1. *All gates $g \in \mathcal{G}$ are in $SU(d)$, that is, they are unitary and have determinant* 1.
2. *For each $g \in \mathcal{G}$ the inverse operation $g^\dagger$ is also in $\mathcal{G}$.*
3. *$\mathcal{G}$ is a universal set for $SU(d)$, i.e., the group generated by $\mathcal{G}$ is dense in $SU(d)$. This means that given any quantum gate $U \in SU(d)$ and any accuracy $\epsilon > 0$ there exists a product $S \equiv g_1 \ldots g_m$ of gates from $\mathcal{G}$ which is an ϵ-approximation to U.*

The notion of approximation being used in this definition, and throughout this paper, is approximation in operator norm. That is, a sequence of instructions generating a unitary operation S is said to be an ϵ-*approximation* to a quantum gate U if $d(U, S) \equiv \|U - S\| \equiv \sup_{\|\psi\|=1} \|(U - S)\psi\| < \epsilon$. In different contexts we will find it convenient to use both the operator norm $\|\cdot\|$, and the associated distance function $d(\cdot, \cdot)$. Another convenient notation is to use S to refer both to a sequence of instructions, $g_1, g_2, \ldots g_m$, and also to the unitary operation $g_1 g_2 \ldots g_m$ corresponding to that sequence. With this convention the "sequence" $g_1, g_2, \ldots$ corresponds to the unitary formed by applying $g_1, g_2, \ldots$ in the *reverse* order. While this is somewhat nonintuitive, in later use it will prove to be the simplest and most natural convention.

We assume the reader is familiar with the elementary properties of the operator norm, which we will mostly use without note. One perhaps somewhat less familiar property we'll have occasion to use is that if H is Hermitian then $d(I, \exp(iH)) = \max_E 2\sin(|E|/2)$, where the maximum is over all eigenvalues E of H. From this it follows easily that $d(I, \exp(iH)) \leq \|H\|$. Furthermore, provided all the eigenvalues are in the range $-\pi$ to π it follows that $d(I, \exp(iH)) = \|H\| + O(\|H\|^3)$.

It is easy to understand the motivation behind parts 1 and 3 of our definition of an instruction set. Less clear is the reason for part 2, namely, the requirement that if $g \in \mathcal{G}$ then we must also have $g^\dagger \in \mathcal{G}$. Our reason for imposing this requirement is that it is used in the proof of the SK theorem, not for any *a priori* reason. In particular, we will make use of the easily-verified fact that if we have an instruction sequence providing an ϵ-approximation to some unitary U, then we can obtain an ϵ-approximation to $U^\dagger$ simply by reversing the order of the sequence, and applying the corresponding inverse instructions. It appears to be an open problem whether a result analogous to the SK theorem holds when condition 2 is relaxed.

The problem of quantum compilation may now be stated more precisely: given an instruction set, $\mathcal{G}$, how may we approximate an arbitrary quantum gate with a sequence of instructions from $\mathcal{G}$; how does the sequence length increase as we require more accurate approximations; and how quickly may we find such an approximating sequence? These final two questions are vitally important, as an inaccurate or inefficient quantum compiler may negate any improvements quantum computers have over classical computers.

As an illustration, suppose we have a quantum algorithm, such as Shor's, which can be broken up into a sequence of m quantum gates, $U_1, \ldots, U_m$. However, let us also suppose that not all of those quantum gates are in our instruction set, and so it is necessary to compile the algorithm so that each gate U_j is re-expressed in terms of the available instruction set. If the compiled algorithm is required to approximate the original to within ϵ, then it is sufficient that each gate U_j be approximated to an accuracy ϵ/m [3]. If we make the *a priori* reasonable assumption [21] that an accuracy of $\delta > 0$ requires an instruction sequence of length $O(1/\delta)$, then each gate U_j will require a sequence of length $O(m/\epsilon)$, and so the total number of instructions required to implement the algorithm is $O(m^2/\epsilon)$ — a quadratic increase in the complexity of the algorithm. For quantum algorithms such as Grover's search algorithm [7, 8], which purports to give a quadratic increase over the best classical algorithm, this may be problematic if the available instruction set does not coincide with the gates used in the algorithm.

The SK theorem may be stated as follows:

Theorem 1 (Solovay-Kitaev) *Let $\mathcal{G}$ be an instruction set for $SU(d)$, and let a desired accuracy $\epsilon > 0$ be given. There is a constant c such that for any $U \in SU(d)$ there exists a finite sequence S of gates from $\mathcal{G}$ of length $O(\log^c(1/\epsilon))$ and such that $d(U, S) < \epsilon$.*

Different proofs of the theorem give different values for c; the proof we describe gives $c \approx 3.97$. We discuss some other proofs (and their corresponding values of c) in Section 6.

A critical issue not addressed in this statement of the SK theorem is the question of how efficiently the approximating sequence S may be found on a classical computer. The SK algorithm both provides a proof of the SK theorem, and also shows that an approximating sequence S may be found efficiently on a classical computer, with a running time of $O(\log^{2.71}(1/\epsilon))$.

At first sight it appears paradoxical that the SK algorithm can produce as output a gate sequence of length $O(\log^{3.97}(1/\epsilon))$ in the asymptotically *shorter* time $O(\log^{2.71}(1/\epsilon))$. We will see that the reason this is possible is because the gate sequence has considerable redundancy, enabling the output to be substantially compressed, i.e., not all the gates need be explicitly output. When this compression is not done, the SK algorithm runs instead in time $O(\log^{3.97}(1/\epsilon))$. Regardless of which method is used, the key point is that the running time is *polylogarithmic* in $1/\epsilon$.

Returning to our earlier example, suppose we have a quantum algorithm expressed in terms of quantum gates $U_1, \ldots, U_m$. If we wish for the compiled algorithm to have accuracy ϵ, then each gate needs accuracy ϵ/m. Using the SK algorithm this can be achieved using a sequence of $O(\log^{3.97}(m/\epsilon))$ quantum gates, and a running time of $O(\log^{2.71}(m/\epsilon))$ on a classical computer. The total number of instructions required to implement the algorithm is therefore $O(m \log^{3.97}(m/\epsilon))$, with an associated classical compile time of $O(m \log^{2.71}(m/\epsilon))$. Thus, the SK algorithm reduces the overhead due to quantum compilation from quadratic to polylogarithmic, which is much more acceptable.

3 The Solovay-Kitaev algorithm for qubits

In this section we present the main ideas used in the SK algorithm. At the conclusion of the section the reader should have a good broad understanding of how and why the algorithm works. Rigorous verification of a few details has been deferred until the next section, in order that those details not obscure the big picture. We also restrict our attention in this section to that form of the SK theorem which applies to *qubits*. The extension to *qudits* involves some new ideas, and is described in Section 5.

The SK algorithm may be expressed in nine lines of pseudocode. We explain each of these lines in detail below, but present it here in its entirety both for the reader's reference, and to stress the conceptual simplicity of the algorithm:

```
function Solovay-Kitaev(Gate $U$, depth $n$)
if ($n == 0$)
  Return Basic Approximation to $U$
else
  Set $U_{n-1}$ = Solovay-Kitaev($U$,$n-1$)
  Set $V$, $W$ = GC-Decompose($UU_{n-1}^\dagger$)
```

```
    Set $V_{n-1}$ = Solovay-Kitaev($V$,$n-1$)
    Set $W_{n-1}$ = Solovay-Kitaev($W$,$n-1$)
    Return $U_n = V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger U_{n-1}$;
```

Let's examine each of these lines in detail. The first line:

```
function Solovay-Kitaev(Gate $U$, depth $n$)
```

indicates that the algorithm is a function with two inputs: an arbitrary single-qubit quantum gate, U, which we desire to approximate, and a non-negative integer, n, which controls the accuracy of the approximation. The function returns a sequence of instructions which approximates U to an accuracy ϵ_n, where ϵ_n is a decreasing function of n, so that as n gets larger, the accuracy gets better, with $\epsilon_n \to 0$ as $n \to \infty$. We describe ϵ_n in detail below.

The `Solovay-Kitaev` function is recursive, so that to obtain an ϵ_n-approximation to U, it will call itself to obtain ϵ_{n-1}-approximations to certain unitaries. The recursion terminates at $n = 0$, beyond which no further recursive calls are made:

```
if ($n == 0$)
    Return Basic Approximation to $U$
```

In order to implement this step we assume that a preprocessing stage has been completed which allows us to find a basic ϵ_0-approximation to arbitrary $U \in SU(2)$. Since ϵ_0 is a constant, in principle this preprocessing stage may be accomplished simply by enumerating and storing a large number of instruction sequences from $\mathcal{G}$, say up to some sufficiently large (but fixed) length l_0, and then providing a lookup routine which, given U, returns the closest sequence. Appropriate values for ϵ_0 and l_0 will be discussed later in this section.

At higher levels of recursion, to find an ϵ_n-approximation to U, we begin by finding an ϵ_{n-1}-approximation to U:

```
else
    Set $U_{n-1}$ = Solovay-Kitaev($U$,$n-1$)
```

We will use U_{n-1} as a step towards finding an improved approximation to U. Defining $\Delta \equiv UU_{n-1}^\dagger$, the next three steps of the algorithm aim to find an ϵ_n-approximation to Δ, where ϵ_n is some improved level of accuracy, i.e., $\epsilon_n < \epsilon_{n-1}$. Finding such an approximation also enables us to obtain an ϵ_n-approximation to U, simply by concatenating our exact sequence of instructions for U_{n-1} with our ϵ_n-approximating sequence for Δ.

How do we find such an approximation to Δ? First, observe that Δ is within a distance ϵ_{n-1} of the identity. This follows from the definition of Δ and the fact that U_{n-1} is within a distance ϵ_{n-1} of U.

Second, decompose Δ as a group commutator $\Delta = VWV^\dagger W^\dagger$ of unitary gates V and W. For any Δ it turns out — this is not obvious — that there is always an infinite set of choices for V and W such that $\Delta = VWV^\dagger W^\dagger$. For our purposes it is important that we find V and W such that $d(I,V), d(I,W) < c_{\mathrm{gc}}\sqrt{\epsilon_{n-1}}$ for some constant c_{gc}. We call such a decomposition a *balanced group commutator*. In Subsection 4.1 we will use the fact that $d(I, \Delta) < \epsilon_{n-1}$ to show that such a balanced group commutator can always be found:

```
Set $V$, $W$ = GC-Decompose($UU_{n-1}^\dagger$)
```

For practical implementations we will see below that it is useful to have $c_{\rm gc}$ as small as possible; the arguments of Subsection 4.1 show that $c_{\rm gc} \approx 1/\sqrt{2}$.

The next step is to find instruction sequences which are ϵ_{n-1}-approximations to V and W:

```
Set $V_{n-1}$ = Solovay-Kitaev($V$,$n-1$)
Set $W_{n-1}$ = Solovay-Kitaev($W$,$n-1$)
```

Remarkably, in Subsection 4.2 we show that the group commutator of V_{n-1} and W_{n-1} turns out to be an $\epsilon_n \equiv c_{\rm approx}\epsilon_{n-1}^{3/2}$-approximation to Δ, for some small constant $c_{\rm approx}$. Provided $\epsilon_{n-1} < 1/c_{\rm approx}^2$, we see that $\epsilon_n < \epsilon_{n-1}$, and this procedure therefore provides an *improved* approximation to Δ, and thus to U. This is surprising, since we are using imperfect approximations to V and W to obtain an improved approximation to the group commutator $VWV^\dagger W^\dagger$. We will see in Subsection 4.2 that the reason this improved approximation is possible is because of cancellation of error terms in the group commutator.

The constant $c_{\rm approx}$ is important as it determines the precision ϵ_0 required of the initial approximations. In particular, we see that for this construction to guarantee that $\epsilon_0 > \epsilon_1 > \ldots$ we must have $\epsilon_0 < 1/c_{\rm approx}^2$. Following the discussion in Subsection 4.2, we obtain $c_{\rm approx} \approx 8c_{\rm gc} \approx 4\sqrt{2}$ and therefore require $\epsilon_0 < 1/32$. Of course, the analysis in Subsection 4.2 simply bounds $c_{\rm approx}$, and it is possible that a more detailed analysis will give better bounds on $c_{\rm approx}$, and thus on ϵ_0. Numerically we have found that for the single-qubit instruction set consisting of the Hadamard gate, the $\pi/8$ gate, and its inverse, $\epsilon_0 \approx 0.14$ and $l_0 = 16$ is sufficient for practical purposes.

The algorithm concludes by returning the sequences approximating the group commutator, as well as U_{n-1}:

```
Return $U_n = V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger U_{n-1}$;
```

Summing up, the function `Solovay-Kitaev(U, n)` returns a sequence which provides an $\epsilon_n = c_{\rm approx}\epsilon_{n-1}^{3/2}$-approximation to the desired unitary U. The five constituents in this sequence are all obtained by calling the function at the $n-1$th level of recursion.

Analysis: We now analyse the runtime of the SK algorithm, and the accuracy of the approximating sequence of instructions. Let l_n be the length of the sequence of instructions returned by `Solovay-Kitaev(U, n)`, and let t_n be the corresponding runtime. Inspection of the pseudocode and the above discussion shows that we have the recurrences

$$\epsilon_n = c_{\rm approx}\epsilon_{n-1}^{3/2} \tag{1}$$
$$l_n = 5l_{n-1} \tag{2}$$
$$t_n \leq 3t_{n-1} + \text{const}, \tag{3}$$

where the constant overhead in t_n comes from the need to perform tasks like finding the balanced group commutator, and other small overheads in the pseudocode. Note that the cost of doing t_n is assumed to come entirely from the calls to `Solovay-Kitaev(., n-1)`, and we have assigned a constant cost for the final line of the algorithm, which returns the new sequence U_n. This assumption is justified if the final line is implemented using pointers to the output from the earlier calls to `Solovay-Kitaev(., n-1)`, rather than returning an actual sequence of gates. This enables us to avoid explicitly returning the (redundant) sequences for $V_{n-1}^\dagger$ and $W_{n-1}^\dagger$. If this is not done, then this analysis needs a little modification; we won't go through the details, but the end result is that the time and length both scale as $O(\log^{3.97}(1/\epsilon))$. Assuming that pointers have been used, the recurrences above imply:

$$\begin{aligned} \epsilon_n &= \frac{1}{c_{\text{approx}}^2} \left(\epsilon_0 c_{\text{approx}}^2\right)^{\left(\frac{3}{2}\right)^n}. & (4)\\ l_n &= O(5^n) & (5)\\ t_n &= O(3^n). & (6)\end{aligned}$$

To obtain a given accuracy ϵ it follows that we must choose n to satisfy the equation:

$$n = \left\lceil \frac{\ln\left[\frac{\ln(1/\epsilon c_{\text{approx}}^2)}{\ln(1/\epsilon_0 c_{\text{approx}}^2)}\right]}{\ln(3/2)} \right\rceil. \tag{7}$$

Substituting this value of n back into Eqs. (5) and (6) we obtain expressions for the length of the approximating sequence and the time of execution, now as a function of ϵ rather than n:

$$\begin{aligned} l_\epsilon &= O\left(\ln^{\ln 5/\ln(3/2)}(1/\epsilon)\right) & (8)\\ t_\epsilon &= O\left(\ln^{\ln 3/\ln(3/2)}(1/\epsilon)\right). & (9)\end{aligned}$$

These are the desired expressions for the length and execution time. Note that the exponent in the first expression is $\ln 5/\ln(3/2) \approx 3.97$, while the exponent in the second expression is $\ln 3/\ln(3/2) \approx 2.71$.

A caveat to this discussion is that we have entirely ignored the *precision* with which arithmetical operations are carried out. In practice we cannot compute with precisely specified unitary operations, since such operations in general require an infinite amount of classical information to specify. Instead, we must work with approximations to such operations, and do all our calculations using finite precision arithmetic. A complete analysis of the SK algorithm should include an account of the cost of such approximations. In practice, we have found that for the values of precision of most interest to us, standard floating-point arithmetic works admirably well, and there seems to be little need for such an analysis.

4 Verification of details

In this section we verify those details of the SK algorithm not fully explained in the previous section. Subsection 4.1 describes how to find balanced group commutators generating a desired unitary, while Subsection 4.2 discusses the error in the group commutator $VWV^\dagger W^\dagger$ when only approximations to the constituent unitaries are available.

4.1 Balanced commutators in $SU(2)$

The earlier discussion of balanced group commutators was in a somewhat specialized notation, which arose out of the application to the Solovay-Kitaev algorithm. To emphasize the generality of the results in this subsection we will use a more generic notation. In particular, suppose $U \in SU(2)$ satisfies $d(I,U) < \epsilon$. Our goal is to find V and W so that the group commutator satisfies $VWV^\dagger W^\dagger = U$, and such that $d(I,V), d(I,W) < c_{\rm gc}\sqrt{\epsilon}$ for some constant $c_{\rm gc}$.

In order to find such a V and W we first examine what seems like a special case. In particular, we choose V to be a rotation by an angle ϕ about the $\hat{x}$ axis of the Bloch sphere, and W to be a rotation by an angle ϕ about the $\hat{y}$ axis of the Bloch sphere. The resulting group commutator $VWV^\dagger W^\dagger$ is a rotation about some axis $\hat{n}$ on the Bloch sphere, by an angle θ satisfying:

$$\sin(\theta/2) = 2\sin^2(\phi/2)\sqrt{1-\sin^4(\phi/2)}. \tag{10}$$

There are many ways of verifying Eq. (10), including simple brute force calculation, perhaps with the aid of a computer mathematics package. One somewhat more elegant way of verifying Eq. (10) is as follows. First, observe that $V^\dagger$ is a rotation by an angle ϕ about the $-\hat{x}$ axis, and thus $WV^\dagger W^\dagger$ must be a rotation by an angle ϕ about the axis $\hat{m}$ that results when $-\hat{x}$ is rotated by an angle ϕ about the $\hat{y}$ axis. That is, $\hat{m}(-\cos(\phi), 0, \sin(\phi))$. The commutator $V(WV^\dagger W^\dagger)$ is thus the composition of these two rotations. The resulting angle of rotation is easy to calculate (see, e.g., Exercise 4.15 on page 177 of [20]), with Eq. (10) the result after some simplification. The exact form of the axis of rotation, $\hat{n}$, is not so important for our purposes, but it is easily computed using similar techniques.

We can easily invert this construction. Suppose now that U is a rotation by an arbitrary angle θ about an arbitrary axis $\hat{p}$ on the Bloch sphere. Define ϕ to be a solution to Eq. (10) for the given value of θ, and define V and W to be rotations by ϕ about the $\hat{x}$ and $\hat{y}$ axes of the Bloch sphere. Then it is easy to see that U must be conjugate to a rotation by θ about the $\hat{n}$ axis identified in the previous two paragraphs, i.e., $U = S(VWV^\dagger W^\dagger)S^\dagger$, for some easily-computed unitary S. It follows that

$$U = \tilde{V}\tilde{W}\tilde{V}^\dagger\tilde{W}^\dagger, \tag{11}$$

where $\tilde{V} \equiv SVS^\dagger$ and $\tilde{W} \equiv SWS^\dagger$. Thus, we can express any unitary U rotating the Bloch sphere by an angle θ as the group commutator of unitaries V and W which rotate the Bloch sphere by an angle ϕ, providing θ and ϕ are related by Eq. (10).

To conclude the argument, note that for a unitary T rotating the Bloch sphere by an angle τ, the distance to the identity satisfies $d(I,T) = 2\sin(\tau/4) = \tau/2 + O(\tau^3)$. Combining this observation with Eq. (10), we see that for U near the identity we can express U in terms of a group commutator of V and W satisfying $d(I,U) \approx 2d(I,V)^2 = 2d(I,W)^2$. That is, we have:

$$\begin{aligned} U &= VWV^\dagger W^\dagger \qquad (12)\\ d(I,V) = d(I,W) &\approx \sqrt{\frac{d(I,U)}{2}} < \sqrt{\frac{\epsilon}{2}}. \qquad (13)\end{aligned}$$

This is the desired balanced group commutator, and gives $c_{\rm gc} \approx 1/\sqrt{2}$. This argument can also easily be modified to give a more rigorous bound on c_{gc}. The details are tedious, but easy to supply, and so we won't do this here.

4.2 Approximating a commutator

Just as in the last subsection, for clarity we state the main result of this subsection in general terms, rather than in the special notation used in Section 3.

Lemma 1 *Suppose $V, W, \tilde{V}$, and $\tilde{W}$ are unitaries such that $d(V,\tilde{V}), d(W,\tilde{W}) < \Delta$, and also $d(I,V), d(I,W) < \delta$. Then:*

$$d(VWV^\dagger W^\dagger, \tilde{V}\tilde{W}\tilde{V}^\dagger\tilde{W}^\dagger) < 8\Delta\delta + 4\Delta\delta^2 + 8\Delta^2 + 4\Delta^3 + \Delta^4. \tag{14}$$

In applying this lemma to the SK algorithm, we replace Δ by ϵ_{n-1}, and δ by $c_{\mathrm{gc}}\sqrt{\epsilon_{n-1}}$. This gives rise to the inequality:

$$d(VWV^\dagger W^\dagger, \tilde{V}\tilde{W}\tilde{V}^\dagger\tilde{W}^\dagger) < c_{\mathrm{approx}}\epsilon_{n-1}^{3/2}, \tag{15}$$

where $c_{\mathrm{approx}} \approx 8c_{\mathrm{gc}}$. With a little more detailed work we can modify this argument to give a rigorous upper bound on c_{approx}. However, just as in the last subsection, the details of this argument are tedious and not especially enlightening, but easy to supply, and so we won't do this here.

Proof: We begin by writing $\tilde{V} = V + \Delta_V, \tilde{W}W + \Delta_W$, which gives:

$$\begin{aligned}\tilde{V}\tilde{W}\tilde{V}^\dagger\tilde{W}^\dagger &= VWV^\dagger W^\dagger + \Delta_V WV^\dagger W^\dagger + V\Delta_W V^\dagger W^\dagger + VW\Delta_V^\dagger W^\dagger + VWV^\dagger\Delta_W^\dagger \\ &\quad + O(\Delta^2) + O(\Delta^3) + O(\Delta^4).\end{aligned} \tag{16}$$

The $O(\Delta^2)$ terms are terms like $\Delta_V\Delta_W V^\dagger W^\dagger$. There are $\binom{4}{2} = 6$ such terms. The $O(\Delta^3)$ terms are terms like $\Delta_V\Delta_W\Delta_V^\dagger W^\dagger$, of which there are $\binom{4}{3} = 4$ terms. There is just a single $O(\Delta^4)$ term, $\Delta_V\Delta_W\Delta_V^\dagger\Delta_W^\dagger$. It follows from these observations, Eq. (16), and the triangle inequality that:

$$\begin{aligned}d(VWV^\dagger W^\dagger, \tilde{V}\tilde{W}\tilde{V}^\dagger\tilde{W}^\dagger) &< \|\Delta_V WV^\dagger W^\dagger + V\Delta_W V^\dagger W^\dagger + VW\Delta_V^\dagger W^\dagger + VWV^\dagger\Delta_W^\dagger\| \\ &\quad + 6\Delta^2 + 4\Delta^3 + \Delta^4.\end{aligned} \tag{17}$$

To complete the proof it suffices to prove that $\|\Delta_V WV^\dagger W^\dagger + VW\Delta_V^\dagger W^\dagger\| < \Delta^2 + 4\Delta\delta + 2\Delta\delta^2$ and $\|V\Delta_W V^\dagger W^\dagger + VWV^\dagger\Delta_W^\dagger\| < \Delta^2 + 4\Delta\delta + 2\Delta\delta^2$. The proofs of the two results are analogous, with the roles of V and W interchanged, and so we only provide the details of the first proof. We expand $W = I + \delta_W$, so that

$$\Delta_V WV^\dagger W^\dagger + VW\Delta_V^\dagger W^\dagger\Delta_V V^\dagger + V\Delta_V^\dagger + O(\Delta\delta) + O(\Delta\delta^2). \tag{18}$$

In this expression the $O(\Delta\delta)$ terms are terms like $\Delta_V\delta_W V^\dagger W^\dagger$. By inspection we see that there are four such terms. The $O(\Delta\delta^2)$ terms are terms like $\Delta V\delta_W V^\dagger\delta_W^\dagger$, and, again by inspection, we see that there are two such terms. It follows that:

$$\|\Delta_V WV^\dagger W^\dagger + VW\Delta_V^\dagger W^\dagger\| < \|\Delta_V V^\dagger + V\Delta_V^\dagger\| + 4\Delta\delta + 2\Delta\delta^2. \tag{19}$$

The unitarity of V and $V + \Delta_V$ implies that $\Delta_V V^\dagger + V\Delta_V^\dagger = -\Delta_V\Delta_V^\dagger$. Combining these observations and using the triangle inequality we obtain

$$\|\Delta_V WV^\dagger W^\dagger + VW\Delta_V^\dagger W^\dagger\| < \Delta^2 + 4\Delta\delta + 2\Delta\delta^2, \tag{20}$$

which completes the proof of the lemma. □

5 The Solovay-Kitaev algorithm for qudits

In this section we present a generalization of the SK algorithm that can be applied to qudits, i.e., to $d \times d$ unitary gates. The main difference is in the group commutator decomposition. In Subsection 5.1 we give a broad description of the modified algorithm, and in Subsection 5.2 we describe the modified group commutator decomposition.

5.1 The modified algorithm

The modified pseudocode for the qudit SK algorithm is as follows:

```
function Solovay-Kitaev(Gate $U$, depth $n$)
if ($n == 0$)
    Return Basic Approximation to $U$
else
    Set $U_{n-1}$ = Solovay-Kitaev($U$,$n-1$)
    Set $V$, $W$ = GC-Approx-Decompose($UU_{n-1}^\dagger$)
    Set $V_{n-1}$ = Solovay-Kitaev($V$,$n-1$)
    Set $W_{n-1}$ = Solovay-Kitaev($W$,$n-1$)
    Return $U_n = V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger U_{n-1}$;
```

Comparing with the earlier pseudocode for the qubit SK algorithm, we see that the only explicit difference is in the line taking the group commutator:

```
Set $V$, $W$ = GC-Approx-Decompose($UU_{n-1}^\dagger$)
```

Recall that in the qubit algorithm the corresponding line finds a balanced group commutator decomposition, i.e., finds V and W such that (a) $VWV^\dagger W^\dagger = \Delta \equiv UU_{n-1}^\dagger$, and (b) $d(I,V), d(I,W) < c_{\rm gc}\sqrt{\epsilon_{n-1}}$ for some constant $c_{\rm gc}$.

In the qudit algorithm this line finds a balanced group commutator which *approximates* Δ. More precisely, we find V and W such that (a) $d(VWV^\dagger W^\dagger, \Delta) < c_{\rm gc'}\epsilon_{n-1}^{3/2}$, for some constant $c_{\rm gc'}$, and (b) $d(I,V), d(I,W) < c_{\rm gc''}\sqrt{\epsilon_{n-1}}$ for some constant $c_{\rm gc''}$. The explicit procedure for doing this is explained in Subsection 5.2, which shows that $c_{\rm gc'} \approx 4d^{3/4}((d-1)/2)^{3/2}$ and $c_{\rm gc''} \approx d^{1/4}((d-1)/2)^{1/2}$, where d is the dimensionality of the Hilbert space we are working in.

The remaining lines work just as in the qubit algorithm, finding instruction sequences V_{n-1} and W_{n-1} which are ϵ_{n-1}-approximations to Δ, and then returning the group commutator corresponding to those sequences, together with the sequence for U_{n-1}, as the output of the algorithm:

```
Set $V_{n-1}$ = Solovay-Kitaev($V$,$n-1$)
Set $W_{n-1}$ = Solovay-Kitaev($W$,$n-1$)
Return $U_n = V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger U_{n-1}$;
```

Although the steps are the same, the analysis is a little different, due to the fact that the group commutator of V and W only approximates Δ. In particular, we have

$$\begin{aligned} d(V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger, \Delta) &\leq d(V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger, VWV^\dagger W^\dagger) + d(VWV^\dagger W^\dagger, \Delta) \quad (21) \\ &< (c_{\mathrm{gc}'} + c_{\mathrm{approx}})\,\epsilon_{n-1}^{3/2}, \quad (22) \end{aligned}$$

where c_{approx} is the same constant that arose in the qubit algorithm (Subsection 4.2), where we estimated $c_{\mathrm{approx}} \approx 8c_{\mathrm{gc}''}$, in the notation of the current section. Thus, just as for the qubit algorithm, the qudit algorithm returns a sequence U_n which provides an $\epsilon_n = O(\epsilon_{n-1}^{3/2})$-approximation to the desired unitary, U. Furthermore, the five constituents in the sequence are all obtained by calling the function at the $n-1$th level of recursion, just as in the qubit algorithm.

The analysis of the accuracy and running time of the qudit SK algorithm proceeds in a fashion analogous to the qubit algorithm, giving $l_\epsilon = O\left(\ln^{\ln 5/\ln(3/2)}(1/\epsilon)\right)$ and $t_\epsilon = O\left(\ln^{\ln 3/\ln(3/2)}(1/\epsilon)\right)$. An important practical caveat concerns the difficulty of constructing the lookup table used to obtain the basic ϵ_0-approximations. This is done by enumerating all possible gate sequences up to some sufficient length. $SU(d)$ is a manifold of dimension d^2-1, so if we wish to approximate every gate in $SU(d)$ to within ϵ_0 then we generate $O(1/\epsilon_0^{d^2-1})$ sequences. For an instruction set $\mathcal{G}$ there are $O(|\mathcal{G}|^l)$ sequences of length $\leq l$, some fraction of which may be redundant. We will therefore need to enumerate all sequences up to a length l_0 satisfying

$$l_0 \geq O\left(\frac{d^2-1}{\log|\mathcal{G}|}\log(1/\epsilon_0)\right). \quad (23)$$

The complexity of the enumeration is exponential in l_0 and thus scales quite poorly with d. While our algorithm is practical for the small values of d of most interest in applications to fault-tolerance (e.g., $d = 2, 3, 4$), it will require great computational effort to scale it to substantially larger values of d.

5.2 Balanced commutators in $SU(d)$

In this subsection our goal is to show that if U satisfies $d(I,U) < \epsilon$ then there exist V and W satisfying $d(VWV^\dagger W^\dagger, U) < c_{\mathrm{gc}'}\epsilon^{3/2}$ and such that $d(I,V), d(I,W) < c_{\mathrm{gc}''}\sqrt{\epsilon}$, for some constants $c_{\mathrm{gc}'}$ and $c_{\mathrm{gc}''}$. The proof combines two lemmas.

Lemma 2 *(Based on Theorem 4.5.2 on page 288 of [12]) Let H be a traceless d-dimensional Hermitian matrix. Then we can find Hermitian F and G such that:*

$$\begin{aligned} [F,G] &= iH \quad (24) \\ \|F\|, \|G\| &\leq d^{1/4}\left(\frac{d-1}{2}\right)^{1/2}\sqrt{\|H\|}. \quad (25) \end{aligned}$$

A variant of this lemma may be found in Problem 8.15 on page 79 of [15]. The variant in [15] has the advantage that the prefactor to $\sqrt{\|H\|}$ on the right-hand side of Eq. (25) is replaced by a constant that does not depend on d. We use the present version since the proof is a little simpler, and it suffices to establish the correctness of the SK algorithm. Readers serious about optimizing this aspect of the SK algorithm should consult [15] for details.

Proof: It is convenient to work in a basis which is Fourier conjugate to the basis in which H is diagonal. (This can, of course, be done, since the commutator bracket is preserved under conjugation, i.e., $S[A,B]S^\dagger = [SAS^\dagger, SBS^\dagger]$ for any matrices A and B, and any unitary S.) In this basis H has the representation

$$H = W \text{diag}(E_1, \ldots, E_d) W^\dagger, \tag{26}$$

where $E_1, \ldots, E_d$ are the eigenvalues of H, and W is the Fourier matrix, with elements $W_{jk} \equiv \omega^{jk}/\sqrt{d}$, where $\omega \equiv \exp(2\pi i/d)$ is a dth root of unity. From Eq. (26) we see that in this basis the diagonal matrix elements of H vanish:

$$H_{jj} = \frac{\sum_k \omega^{jk} E_k \omega^{jk*}}{d} = \frac{\sum_k E_k}{d} = \frac{\text{tr}(H)}{d} = 0. \tag{27}$$

As a trial solution to the equation $[F,G] = iH$ we will assume that G is some diagonal matrix, with real entries. The condition $[F,G]iH$ is then equivalent to $iH_{jk} = F_{jk}(G_{kk} - G_{jj})$. This suggests imposing the condition that the diagonal entries of G all be distinct, and defining

$$F_{jk} \equiv \begin{cases} \frac{iH_{jk}}{G_{kk} - G_{jj}} & \text{if } j \neq k; \\ 0 & \text{if } j = k. \end{cases} \tag{28}$$

It is easy to see that with these choices F and G are Hermitian, and satisfy $[F,G] = iH$.

What of the norms $\|F\|$ and $\|G\|$? Suppose we choose the diagonal entries of G as $-(d-1)/2, -(d-1)/2 + 1, \ldots, (d-1)/2$. With this choice it is clear from our definition that the entries of F satisfy $|F_{jk}| \leq |H_{jk}|$, and we have

$$\begin{aligned} \|F\|^2 &\leq \text{tr}(F^2) & (29) \\ &\leq \text{tr}(H^2) & (30) \\ &\leq d\|H\|^2, & (31) \end{aligned}$$

where the first and third inequalities follow easily from the definitions, and the second inequality follows from the fact that $|F_{jk}| \leq |H_{jk}|$. With these choices we therefore have $\|F\| \leq \sqrt{d\|H\|}$ and $\|G\| = (d-1)/2$. Rescaling F and G appropriately we can ensure that the equation $[F,G] = iH$ remains satisfied, while satisfying Eq. (25). This completes the proof. □

Lemma 3 *Suppose F and G are Hermitian matrices such that $\|F\|, \|G\| < \delta$. Then:*

$$d\left(\exp(iF)\exp(iG)\exp(-iF)\exp(-iG), \exp(i \times i[F,G])\right) \leq c_1 \delta^3, \tag{32}$$

for some constant $c_1 \approx 4$.

Proof: This is easily verified using the standard series expansion for matrix exponentials. Alternately, this is also a standard result in the theory of Lie groups. See, e.g., Proposition 2 on page 25 in Section 1.3 of [22]. □

The result we desire may be obtained by combining these two lemmas. Suppose $d(I,U) < \epsilon$. We can find Hermitian H such that $U \exp(iH)$ and $d(I,U) = \|H\| + O(\|H\|^3)$. By Lemma 2 we can find Hermitian F and G such that $[F,G] = iH$, and $\|F\|, \|G\| \leq c_{\text{gc}''}\sqrt{\epsilon}$

for $c_{\mathrm{gc}''} \approx d^{1/4}\sqrt{(d-1)/2}$. Setting $V \equiv \exp(iF), W \equiv \exp(iG)$, $\delta \equiv c_{\mathrm{gc}''}\sqrt{\epsilon}$ and applying Lemma 3, we obtain:

$$d(VWV^\dagger W^\dagger, U) < c_1(c_{\mathrm{gc}''}\sqrt{\epsilon})^3 = c_{\mathrm{gc}'}\epsilon^{3/2}, \tag{33}$$

where $c_{gc'} = c_1 c_{\mathrm{gc}''}^3 \approx 4d^{3/4}((d-1)/2)^{3/2}$. Furthermore, we have $d(I,V), d(I,W) < c_{\mathrm{gc}''}\sqrt{\epsilon}$, as desired.

6 Prior work

The treatment of the SK theorem and algorithm given in this paper is based upon Appendix 3 in [20]. That appendix gave a detailed description of the SK theorem, and included an exercise (Exercise A3.6) asking the reader to find an algorithm for efficiently finding accurate sequences of instructions approximating a desired unitary. Reader feedback on [20] suggests that this was not one of the easier exercises in that volume, a fact that partially inspired the present review paper.

The history of the SK theorem and algorithm is interesting. For $SU(2)$ the SK theorem was announced by Solovay in 1995 on an email discussion list, but no paper has subsequently appeared. Independently, in 1997 Kitaev [13] outlined a proof in a review paper, for the general case of $SU(d)$. After hearing of this result, Solovay announced that he had generalized his proof in a similar fashion. Kitaev's paper also addressed the question of efficient implementation, sketching out an algorithm to quickly find accurate instruction sequences approximating a desired gate. The proof of the SK theorem given in [20] was based on [13], discussions with Kitaev, Solovay, and Freedman, and on a 1999 lecture by Freedman, based on Kitaev's proof.

Kitaev's 1997 discussion [13] of the SK theorem and algorithm is described and extended in the 2002 text by Kitaev, Shen and Vyalyi [15]. Indeed, that text actually presents two different approaches to the problem. The first approach, described in Section 8.3 of [15], uses very similar ideas to the description we have given, with a few technical differences resulting in somewhat different behaviour for the algorithm's running time, and for the length of the instruction sequences produced. In particular, the algorithm in [15] has a running time $O(\log^{3+\delta}(1/\epsilon))$ and produces an instruction sequence of length $O(\log^{3+\delta}(1/\epsilon)$, where δ can be chosen to be any positive real number.

The second approach, described in Section 13.7 of [15], is quite different in flavour. It modifies our setting for the SK theorem and algorithm in two ways: (1) it makes use of ancilla qubits, and (2) it constrains the allowed instruction set somewhat, for example, to Clifford group gates, together with the Toffoli gate; certain other instruction sets are also possible, but not an arbitrary instruction set. A variant of Kitaev's phase estimation algorithm [16, 14] is used to construct instruction sequences providing exponentially accurate conditional phase shifts. Standard constructions (e.g. [2]), together with these phase shift gates, can then be used to approximate the desired gates. The running time for this approach is $O(\log^2(1/\epsilon)\log(\log(1/\epsilon)))$, and the length of the instruction sequence is $O(\log^2(1/\epsilon)\log(\log(1/\epsilon)))$. This approach also has the advantage of being easily parallelizable, when acting on multiple qubits.

A non-constructive approach to the SK theorem was taken by Harrow, Recht and Chuang in [9], based on Harrow's undergraduate thesis [10] and papers by Arnold and Krylov [1], and by Lubotsky, Phillips and Sarnak [18, 19] (c.f. [6]). In particular, [9] proves that for a suitable choice of instruction set, an approximation of accuracy ϵ may be achieved using a sequence of $O(\log(1/\epsilon))$ instructions. It does not yet seem to be well understood exactly which instruction sets achieve such a fast rate of convergence. Furthermore, [9] describe a simple volume argument establishing that, up to a constant factor, it is not possible to obtain shorter approximating sequences, and so this result is optimal. However, they do not provide a constructive means of efficiently finding these short instruction sequences.

We conclude by noting two papers that, while not directly concerned with the SK theorem or algorithm, are likely of interest to anyone interested in quantum compilation. The first is a paper by Freedman, Kitaev, and Lurie [5], who develop some general conditions under which a subset $\mathcal{G}$ of a semisimple Lie group G must generate a dense subset of G. In particular, they introduce a natural family of metrics defined for any semisimple Lie group G, and show that there is a *universal* constant c, independent of G, so that provided all points in G are within a distance c of $\mathcal{G}$, then $\mathcal{G}$ must generate a dense subset of G. Roughly speaking, provided $\mathcal{G}$ "fills" G sufficiently well, it is guaranteed to generate a dense subset of G.

The second paper of interest is by Fowler [4], who investigates the feasibility of compiling Shor's algorithm into fault-tolerant form using a search technique that, while while not "efficient" in the sense of the SK algorithm, in practice seems to yield promising results.

7 Conclusion

The Solovay-Kitaev theorem and algorithm are fundamental results in the theory of quantum computation, and it seems likely that variants of these results will be used in future implementations of quantum computers to compile quantum algorithms such as Shor's into a fault-tolerant form. The discussion of these results in the present review paper has been pedagogical, aimed at a formulation that brings out the key ideas, rather than being optimized for accuracy and efficiency. It is an interesting open problem to determine the extent to which these constructions can be improved, perhaps even developing a version of the Solovay-Kitaev algorithm achieving optimal or near-optimal accuracy and efficiency.

Acknowledgments

We thank Mike Freedman, Aram Harrow, and Alexei Kitaev for enlightening discussions about the Solovay-Kitaev theorem and algorithm, and Mark de Burgh, Mark Dowling, and Yeong Cherng Liang for helpful comments on a draft of the paper.

References

1. V. I. Arnold and A. L. Krylov. *Soviet Math. Dokl.*, 4:1, 1962.
2. A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, 1995. arXiv:quant-ph/9503016.
3. E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comp.*, 26(5):1411–1473, 1997. arXiv:quant-ph/9701001.
4. A. G. Fowler. Constructing arbitrary single-qubit fault-tolerant gates. *arXiv:quant-ph/0411206*, 2004.
5. M. Freedman, A. Kitaev, and J. Lurie. Diameters of homogeneous spaces. *arXiv:quant-ph/0209113*, 2002.

6. A. Gamburd, D. Jakobson, and P. Sarnak. Spectra of elements in the group ring of $su(2)$. *J. Eur. math. Soc.*, 1(1):51–85, 1999.
7. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM Symposium on Theory of Computation*, page 212, New York, 1996. Association for Computing Machinery.
8. Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79(2):325, 1997. arXiv:quant-ph/9706033.
9. A. Harrow, B. Recht, and I. L. Chuang. Efficient discrete approximations of quantum gates. *J. Math. Phys.*, 43:4445, 2002. arXiv:quant-ph/0111031.
10. A. W. Harrow. Quantum compiling. MIT undergraduate thesis, 2001.
11. R. R. Tucci. A rudimentary quantum compiler. arXiv:quant-ph/9805015, 1998.
12. R. A. Horn and C. R. Johnson. *Topics in matrix analysis.* Cambridge University Press, Cambridge, 1991.
13. A. Y. Kitaev. Quantum computations: algorithms and error correction. *Russ. Math. Surv.*, 52(6):1191–1249, 1997.
14. A. Y. Kitaev. Quantum error correction with imperfect gates. In A. S. Holevo O. Hirota and C. M. Caves, editors, *Quantum Communication, Computing, and Measurement*, pages 181–188, New York, 1997. Plenum Press.
15. A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and quantum computation*, volume 47 of *Graduate Studies in Mathematics.* American Mathematical Society, Providence, Rhode Island, 2002.
16. A. Yu. Kitaev. Quantum measurements and the Abelian stabilizer problem. *arXiv:quant-ph/9511026,*, 1995.
17. D. E. Knuth. Computer science and its relation to mathematics. *Amer. Math. Month.*, 81(4), April 1974.
18. A. Lubotsky, R. Phillips, and P. Sarnak. Hecke operators and distributing points on the sphere i. *I. Comm. Pure Appl. Math.*, 39:S149–S186, 1986.
19. A. Lubotsky, R. Phillips, and P. Sarnak. Hecke operators and distributing points on the sphere ii. *I. Comm. Pure Appl. Math.*, 40:401–420, 1987.
20. M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information.* Cambridge University Press, Cambridge, 2000.
21. J. Preskill. Reliable quantum computers. *Proc. Roy. Soc. A: Math., Phys. and Eng.*, 454(1969):385–410, 1998.
22. W. Rossmann. *Lie Groups: An Introduction Through Linear Groups.* Oxford University Press, Oxford, 2002.
23. P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings, 35th Annual Symposium on Fundamentals of Computer Science*, Los Alamitos, 1994. IEEE Press.
24. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comp.*, 26(5):1484–1509, 1997.