Efficiently Computing the Jones Polynomial on a Quantum Computer

Bibek Pokharel, Ross Jennings, Kenneth Schiller

May 25, 2015

Abstract

These lectures notes are based on the paper “A Polynomial Quantum Algorithm for Approximating the Jones Polynomial” published in 2008 by Dorit Aharanov, Vaughan Jones, and Zeph Landau. The aim of these notes is to present their algorithm with enough background information so that it can be understood by someone who does not have a background in knot theory or quantum computing. We begin by describing the braid representation of knots and the use of the Jones polynomial as a knot invariant. We then use the Temperley-Lieb Algebra $TL_n$ to describe a matrix representation of braids, and show that the Jones polynomial of a knot can be calculated by taking a trace of the corresponding matrix. We introduce the circuit model of quantum computing and finally present the quantum algorithm for computing the Jones polynomial, with discussion about how to understand the “quantum speedup” achieved by this algorithm.
1 Braids and Knots

(See chapter 1 of [1] for knots, links, and tricolorability, and section 5.4 of [1] for braids and braid words.)

1.1 What Are Knots?

Definition 1.1. A knot is a closed curve in $S^3$ that does not intersect itself anywhere.

- Compare with everyday definition of knots

Definition 1.2. Two knots $K_1$ and $K_2$ are equivalent if their complements $S^3\setminus K_1$ and $S^3\setminus K_2$ are isotopic.

Intuitive Definition: Two knots are equivalent ("the same") if one can be continuously deformed into the other. A deformation means bending and stretching the knot, without cutting, and without allowing the knot to pass through itself.

Definition 1.3. A knot diagram is a kind of picture of a knot that is a projection of the knot onto a plane. The points where the knot crosses over itself are called crossings. In the crossings of a diagram we must indicate which strand passes over which.

A diagram with zero crossings is essentially a circle. This is the simplest knot, and it is called the unknot, or the trivial knot. Note that any diagram with a single crossing is also the unknot.

Figure 1: All four configurations of a knot diagram with one crossing.

It is easy to see that these knots are trivial. In fact, the simplest nontrivial knot has three crossings. But how do we know this for sure? How do we even know that there exists any nontrivial knot? More generally, we would like to know when two diagrams represent either the same or different knots.

Theorem 1.1.

Two diagrams represent the same knot iff they are related by a sequence of Reidemeister moves.

There are three types of Reidemeister move:

- Type I

- Type II
Definition 1.4. A link is a set of knots which may be tangled up together. Each knot is called a component of the link.

Here are two simple links of two components: the 2-unlink and the Hopf link.

Here is a link of three components called the Borromean rings.

We talk about links much in the same way we talk about knots. For example:

- Two links are the same if we can deform one into the other.
- The Reidemeister moves apply to links in the same way as to knots.

1.2 Knot and Link Invariants

Definition 1.5. A knot (or link) invariant is a property we can calculate from a diagram which will have the same value for any two diagrams which represent the same knot (or link). An invariant helps us to prove that two diagrams represent different knots (or links).

We can prove something is a link invariant by showing that it remains constant across each of the Reidemeister moves.

Definition 1.6. A diagram is tricolorable if it is possible to color each of its strands with one of three colors such that at least two colors appear in the diagram and at each crossing, either all three colors appear or only a single color appears.

Examples of “good” and “bad” crossings, in the sense that a diagram is tricolorable iff it can be colored with all good crossings.

Theorem 1.2. Tricolorability is a link invariant.

For a sketch of the proof, see pages 24–25 in The Knot Book [1].

Tricolorability allows us to prove that the trefoil is distinct from both the unknot and the figure-eight. However, it will not let us tell the unknot apart from the figure-eight.
1.3 Braids

A braid is not a knot, but we can use braids to represent knots.

**Definition 1.7.** A braid consists of two horizontal bars, one over the other, with \( n \) marked points, and \( n \) strands which connect points on the top with points on the bottom in a one-to-one correspondence. These strands always head downward (they don't turn back upwards) and they may cross over one another as do the strands in knot diagrams.

Note that:

- We define equivalence on braids in the same way as with knots and links
- The Reidemeister moves apply to braid diagrams in the same ways as to knot diagrams

![Braid Diagrams](image)

(a) A braid  (b) The same braid  (c) NOT a braid

**Figure 3:** Braid examples.

How do we use braids as representations for knots?

**Definition 1.8.** We take the trace closure of a braid by connecting each marked point on the top bar of the braid with its respective point from the bottom bar, with strands which do not cross each other or any part of the braid. We then remove the horizontal bars.

The trace closure of a braid is a knot or a link. Thus we can say that a braid represents a given knot if the trace closure of that braid is equivalent to the knot.

**Theorem 1.3.** Any knot has a braid representation under the trace closure.

For a proof, see pages 129–131 in *The Knot Book* [1].

Given a knot, we can find a braid representation in two ways:

- Give the knot an orientation and find a point on a knot diagram such that the knot is either always traveling clockwise or always traveling counter-clockwise. This is not possible on every diagram.
Figure 4: Conversion from knot to braid and braid to knot, via the trace closure.

- Use the algorithm described in pages 129–131

**Definition 1.9.** Let $a$ and $b$ be $n$-strand braids. Then $a$ composed with $b$ is an $n$-strand braid $ab$ which is taken by stacking $a$ on top of $b$ and removing the horizontal bars where they meet.

**Definition 1.10.** The plat closure of a braid with $2n$ strands is given by connecting the marked points to adjacent points on the same bar, two at a time in consecutive order.

Figure 5: The trace closure(left) and plat closure (right) of the same braid.

1.4 Braid Words

We can reshape a braid such that no two crossings occur at the same height. This means we can think of any $n$-strand braid of $m > 0$ crossings as the composition of $m$ $n$-strand braids, each with a single crossing.

We denote each single-crossing braid by $a_i$ if the left strand crosses over the right (moving from top to bottom) and by $a_i^{-1}$ otherwise, where $i$ is an integer from $1$ to $n-1$ denoting the position of the crossing. We denote the braid with no crossings by $I$. We call these the generators of the braid group of $n$ strands.

**Definition 1.11.** A braid word is a representation of a braid in terms of the symbols of the generators that make up the braid.

This gives us a compact way of describing knots. For example, as illustrated above, we can describe the figure-eight knot by the braid word $\sigma_2\sigma_1^{-1}\sigma_2\sigma_1^{-1}$ with $n = 3$.

Braid words are elements of a group under the composition operation.
• Closure: The composition of two braids is a braid.
• Associativity: For braids $a, b, c$ it is the case that $(ab)c = a(bc)$.
• Identity: Any braid composed with $I$ is itself.
• Inverse: Given some braid, flip it vertically. This is its inverse because composition with the original braid will produce $I$.

Theorem 1.4. Two braid words represent the same braid if and only if we can go from one to the other using the following rules:

1. $\sigma_i \sigma_i^{-1} = \sigma_i^{-1} \sigma_i = I$ and $wI = w = Iw$
2. $\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}$
3. $\sigma_i \sigma_j = \sigma_j \sigma_i$ for $|i - j| \geq 2$

Theorem 1.5. Two braid words represent the same oriented link if and only if we can go from one to the other using the three rules above and the two rules below:

4. $w \rightarrow \sigma_i w \sigma_i^{-1}$ or $w \rightarrow \sigma_i^{-1} w \sigma_i$ (Conjugation)
5. $w \rightarrow w \sigma_n$ or $w \rightarrow w \sigma_n^{-1}$ if $w$ is an $n$-strand braid. (Stabilization)

(a) Conjugation  
(b) Stabilization

Figure 7: Visualizations of the conjugation and stabilization rules.

Exercise 1

1. Draw the braid corresponding to the braid word $\sigma_3 \sigma_1^{-1} \sigma_2 \sigma_3^{-1}$.
2. Write the braid word for the figure below and then simplify it using the rules in Theorem 1.4.
2 The Jones Polynomial

One particularly interesting and powerful example of a knot invariant is the Jones polynomial. The Jones polynomial \( V(K) \) of a knot \( K \) is a Laurent polynomial in the variable \( A \) with integer coefficients. A Laurent polynomial in \( A \) differs from an ordinary polynomial in \( A \) in that it can have terms with negative powers of \( A \). For example, \( A^3 - 1, -A^2 - A^{-2}, \) and \( A^{-1} + 2A^{-5} - A^{-9} \) are all Laurent polynomials in \( A \), but only the first is an ordinary polynomial in \( A \).

The Jones polynomial can be computed using a function of knot diagrams known as the Kauffman bracket.

**Definition 2.1.** The **Kauffman bracket** of a knot diagram \( D \) is denoted \( \langle D \rangle \) and can be computed recursively using the following relations, known as *skein relations*:

\[
\begin{align*}
\langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle &= A \langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle + A^{-1} \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle \\
\langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle &= (A^2 - A^{-2}) \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle \\
\langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle &= 1.
\end{align*}
\]  

The first two relations are meant to be applied locally – the first can be applied to any single crossing in a diagram, and the second can be applied to any closed loop with no crossings that can be separated from the rest of the diagram. By successively applying the first skein relation (2.1) to each crossing in the diagram, the bracket of any diagram can be reduced to a linear combination of brackets of diagrams with no crossings. The second skein relation (2.2) can then be applied to remove circles from these diagrams until only one circle is left in each. Finally, the last skein relation (2.3) provides a way to do away with the diagrams altogether, so that only a polynomial is left.

The Kauffman bracket of the trefoil knot, for instance, is as follows:

\[
\langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle = A \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle + A^{-1} \langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle \\
= A \left( A \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle + A^{-1} \langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle \right) + A^{-1} \left( A \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle + A^{-1} \langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle \right) \\
= A^2 \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle + (2 + A^{-2}(A^2 - A^{-2})) \langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle \\
= A^2 \left( A \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle + A^{-1} \langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle \right) + (2 - 1 - A^{-4}) \left( A \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle + A^{-1} \langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle \right) \\
= A^3(-A^2 - A^{-2}) + A + (1 - A^{-4})(A + A^{-1}(-A^2 - A^{-2})) \\
= -A^5 - A + A + (1 - A^{-4})(A - A - A^{-3}) \\
= -A^5 - A^3 + A^{-7}.
\]

Unfortunately, the Kauffman bracket on its own is not a knot or link invariant. It is invariant under Type II and III Reidemeister moves, but not Type I moves. In particular,

\[
\langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle = A \langle \begin{array}{c}
\circlearrowleft \\
\end{array} \rangle + A^{-1} \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle = (A + A^{-1}(-A^2 - A^{-2})) \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle = -A^{-3} \langle \begin{array}{c}
\circlearrowright \\
\end{array} \rangle.
\]

This suggests a way to “fix” the Kauffman bracket to create a knot invariant.
Definition 2.2. The **writhe** \( \text{wr}(D) \) of a knot diagram \( D \) is an integer which can be calculated from a knot diagram as follows:

First, orient the knot diagram by choosing one of the two possible directions to go around it. Each crossing can now be categorized as either positive or negative, depending on the relative orientation of the upper and lower strands. The writhe is then the number of positive crossings minus the number of negative crossings. It turns out that the resulting number is independent of the orientation chosen for the knot, and that it is also invariant under Type II and III Reidemeister moves, and changes by \( \pm 1 \) under a Type I Reidemeister move (see Exercise 2).

Because of this, it is possible to combine the Kauffman bracket with the writhe to create a knot invariant.

Definition 2.3. The **Jones polynomial** \( V(D) \) of a knot diagram \( D \) is given by

\[
V(D) = (-A^3)^{-\text{wr}(D)} \langle D \rangle.
\]

(2.4)

Theorem 2.1. The Jones polynomial is an invariant of knots. That is, if \( D_1 \) and \( D_2 \) are two diagrams representing the same knot, then \( V(D_1) = V(D_2) \).

Proof. Suppose \( D_1 \) and \( D_2 \) are two diagrams representing the same knot \( K \). By Theorem 1.1, there is a sequence of Reidemeister moves that takes \( D_1 \) to \( D_2 \). Applying Type II and III Reidemeister moves doesn’t change the Kauffman bracket or the writhe, but applying a Type I Reidemeister move to some intermediate diagram \( D \) takes \( \langle D \rangle \) to \( -A^3 \langle D \rangle \) and \( \text{wr}(D) \) to \( \text{wr}(D) + 1 \). This means that the new value of the Jones polynomial is

\[
V(D') = (-A^3)^{-\text{wr}(D')}(D')
= (-A^3)^{-\text{wr}(D)-1}(-A^3 \langle D \rangle)
= (-A^3)^{-\text{wr}(D)} \langle D \rangle = V(D).
\]

Therefore \( V(D) \) is also invariant under Type I Reidemeister moves. This means that the entire sequence of moves has no effect on the Jones polynomial, so \( V(D_1) = V(D_2) \).

The Jones polynomial has the interesting property that all of the terms have exponents that are multiples of 4 (see Exercises 4 and 5). Because of this, it is often written in terms of the variable \( t = A^{-4} \).

Exercises

1. Show that the Kauffman bracket is invariant under Type II and Type III Reidemeister moves.
2. Show that the writhe of a diagram is invariant under Type II and III Reidemeister moves, and that it changes by \( \pm 1 \) under Type I moves.
3. Compute the Jones polynomial of the knot represented by the diagram shown below.

4. Show that the exponents of any two nonzero terms in the Kauffman bracket of a knot diagram differ by a multiple of 4.
5. Using the result of the last exercise, show that the exponent of any nonzero term in the Jones polynomial is a multiple of 4. (Hint: Look at the term corresponding to the highest possible power of \( A \).)
3 Temperley-Lieb Algebra

It is also possible to arrive at the Jones polynomial by applying the Kauffman bracket skein relations (2.1-2.3) to braids. Since braids are not closed, however, the pictures we get by resolving all the crossings using the first relation do not all consist of disjoint circles. Specifically, we can define the picture \( E_i \) as follows.

\[
E_i = AE_i + A^{-1}E_0
\] (3.1)
\[
E_i^{-1} = AE_0 + A^{-1}E_i
\] (3.2)

In general, the results of applying Equation 2.1 to a braid will be a linear combination of pictures without crossings, which, like braids, each consist of two horizontal bars, each with \( n \) marked points, which are connected by strands. However, unlike what is the case with braids, these strands may turn back upwards or connect points which are on the same side of the diagram.

We can multiply these pictures in the same way we multiply braids – by stacking them. This multiplication can also be extended linearly, so that we can form products of arbitrary linear combinations of pictures. In addition to counting arbitrary deformations of the pictures as equivalent, we will apply the second Kauffman bracket skein relation (Equation 2.2) whenever we can to convert closed loops into factors of \(-A^2-A^{-2}\). Because they can be added, multiplied, and multiplied by scalars, these linear combinations of pictures form a structure known as an algebra.

Definition 3.1. An (associative) algebra \( A \) over a field \( F \) is a vector space over \( F \) together with an operation \( \cdot : A \times A \rightarrow A \) and an element \( 1 \in A \) satisfying

1. \( x \cdot (y + z) = x \cdot y + x \cdot z \)
2. \( (x + y) \cdot z = x \cdot z + y \cdot z \)
3. \( (\lambda x) \cdot (\mu y) = (\lambda \mu)(x \cdot y) \)
4. \( (x \cdot y) \cdot z = x \cdot (y \cdot z) \)
5. \( 1 \cdot x = x \)

for all \( x, y \in A \) and all \( \lambda, \mu \in F \).

That is, an algebra is a vector space whose elements can be multiplied in a way that is compatible with the vector space structure. Note that every algebra over a given field \( F \) is also a ring.

Definition 3.2. The Temperly-Lieb algebra \( TL_n \) is the algebra over \( \mathbb{C} \) generated the pictures \( E_1, E_2, \ldots, E_{n-1} \) under multiplication by stacking. These generators satisfy the relations:

\[
E_iE_j = E_jE_i \text{ for } |i - j| \geq 2
\] (3.3)
\[
E_iE_{i+1}E_i = E_i
\] (3.4)
\[
E_i^2 = (-A^2-A^{-2})E_i
\] (3.5)

Source: [2].
\[ E_i E_j = E_j E_i \quad \text{for} \quad |i-j| \geq 2 \]

\[ E_i E_{i+1} E_i = E_i \]

\[ E_i^2 = (-A^2 - A^{-2})E_i \]

Figure 9: Picture proofs for the three relations that are satisfied by the elements of the Temperly-Lieb algebra.

**Example:** The figure-8 knot can be represented as a braid with braid word \( \sigma_2 \sigma_1^{-1} \sigma_2 \sigma_1^{-1} \). Using TL\(_n\) elements this can be written as

\[ \sigma_2 \sigma_1^{-1} \sigma_2 \sigma_1^{-1} = (AE_2 + A^{-1}E_0) \times (AE_0 + A^{-1}E_1) \times (AE_2 + A^{-1}E_0) \times (AE_0 + A^{-1}E_1) \]

**Definition 3.3.** The Markov Trace is defined on an element \( E_i \) of TL\(_n\) such that \( \text{tr}(E_i) = (-A^{-2}A^2)^{a-n} \), where \( a \) is the number of loops in the trace closure of \( E_i \). The Markov trace satisfies the following three properties:

- \( \text{tr}(E_0) = 1 \)
\begin{itemize}
  \item $\text{tr} (E_i E_j) = \text{tr}(E_i) \text{tr}(E_j)$
  \item If $X \in TL_{n-1}$ then $\text{tr}(XE_{n-1}) = \frac{1}{(-A^{-2} - A^2)} \text{tr}(X)$
\end{itemize}

Source: [2].

**Exercises**

1. Prove using pictures that $\text{tr}(E_0) = 1$, $\text{tr} (E_i E_j) = \text{tr}(E_i) \text{tr}(E_j)$ and if $X \in TL_{n-1}$ then $\text{tr}(XE_{n-1}) = \frac{1}{(-A^{-2} - A^2)} \text{tr}(X)$.
2. Compute the Markov Trace for $E_1 + E_0$.
3. Prove that $\text{tr}(E_1) = (-A^2 - A^{-2})^{-1}$.
4. Expand $\sigma_1 \sigma_1 \sigma_1$ into terms comprising $TL_n$ elements and compute the Markov trace of this braid. Take the trace closure of the original braid, can you identify the knot so formed?
4 The Path Model Representation

(See section 2.13 (pages 410-413) in Aharonov et al. [2] for a discussion of path model representation.)

We can deal with braids in a much more computationally tractable way by representing them as matrices, which describe linear transformations on a vector space. The word “represent” here is used in a technical sense which will be explained shortly. In particular, we will be interested in a particular representation of the braid group, called the path model representation, which is closely connected to the Kauffman bracket and the Jones polynomial.

To begin with, let’s define what we mean by a representation.

**Definition 4.1.** Let $G$ be a group and $V$ be a vector space (over $\mathbb{C}$). A representation of $G$ in $V$ is a group homomorphism $\rho$ from $G$ to the group $GL(V)$ of invertible linear transformations from $V$ to itself. Source: [6].

This definition could stand a little unpacking. First, the set $GL(V)$ of invertible linear transformations from $V$ to itself is in fact a group under the operation of function composition: The identity is invertible, the inverse of an invertible linear transformation is invertible, and the product of two invertible linear transformations is also invertible, with the inverse being given by the “shoes-and-socks” formula: $(TU)^{-1} = U^{-1}T^{-1}$.

Furthermore, with respect to any given basis of $V$, elements of $GL(V)$ can be written as matrices, with matrix multiplication taking the place of function composition. In fact, we can think of $GL(V)$ as a group of $n \times n$ invertible matrices, where $n$ is the dimension of $V$, without losing any information.

Since a representation $\rho$ of $G$ in $V$ is a function from $G$ to $GL(V)$, we can think of it as a way of associating linear transformations (or, for any chosen basis, matrices) to elements of the group. To say that $\rho$ is a group homomorphism means simply that, for any group elements $g, h \in G$,

$$\rho(gh) = \rho(g)\rho(h),$$

where the multiplication on the left is in $G$ and the multiplication on the right is composition (matrix multiplication) in $GL(V)$.

Ultimately, we are going to want to use the Kauffman bracket skein relations to define a representation of the braid group $B_n$ in a finite-dimensional complex vector space. As we have seen before, applying the Kauffman bracket skein relations to a braid in $B_n$ produces an element of the Temperley-Lieb algebra $TL_n$, and we are going to want to use the algebraic structure of $TL_n$ to simplify our definition of the path model representation. To do this, we need to think about the closely related notion of representations of algebras.

**Definition 4.2.** If $A$ is an algebra and $V$ is a vector space over $\mathbb{C}$, a representation of $A$ in $V$ is an algebra homomorphism $\tau$ from $A$ to the algebra $L(V)$ of linear transformations from $V$ to itself. Source: [6].

A representation of an algebra is entirely analogous to a representation of a group. Again, we are dealing with a homomorphism from an abstract algebraic structure to a set of linear transformations, although this time it is a homomorphism of algebras rather than a homomorphism of groups. The set of all linear transformations $L(V)$ from $V$ to itself is in fact an algebra, since linear transformations, in addition to being multiplied, can be added and multiplied by scalars. The group $GL(V)$ sits inside this algebra – in fact, $GL(V)$ is precisely the group of units of the ring $L(V)$.

The fact that an algebra representation $\tau$ is a homomorphism of algebras means that it preserves addition, scalar multiplication, and the multiplication of algebra elements. That is, the following equations hold for all scalars $\lambda$ and $\mu$ and algebra elements $a$ and $b$:

$$\tau(\lambda a + \mu b) = \lambda \tau(a) + \mu \tau(b) \quad (4.1)$$

$$\tau(ab) = \tau(a)\tau(b). \quad (4.2)$$

We already have a function $\phi : B_n \to TL_n$ that preserves multiplication. This means that if we can find a representation $\tau$ of the algebra $TL_n$, we can get a representation $\rho$ of the group $B_n$. In particular, we can define $\rho(g) = \tau(\phi(g))$ for every $g \in B_n$. Our new function $\rho$ preserves multiplication since both $\tau$ and $\phi$ do:

$$\rho(gh) = \tau(\phi(gh)) = \tau(\phi(g)\phi(h)) = \tau(\phi(g))\tau(\phi(h)) = \rho(g)\rho(h).$$
Moreover, this means that \( \rho(g^{-1})\rho(g) = \rho(g^{-1}g) = \rho(e) \), which must be the identity (see Exercise 1), so \( \rho(g) \) is an invertible linear transformation for every \( g \in B_n \). Therefore \( \rho \) is a representation of \( B_n \).

### 4.1 The representation of \( TL_n \)

Before we can say how to find the matrix representing a braid \( b \in B_n \), we must define the vector space \( V \) we are going to represent \( TL_n \) in. Our definition of \( V \) will use the notion of paths on a graph. We can think of a graph as a set of dots (vertices) connected by lines (edges). We will mostly be concerned with the straight-line graph \( \Gamma_{k-1} \) on \( k-1 \) vertices, where \( k \) is a positive integer of our choice. This graph looks something like this:

![Diagram of \( \Gamma_{k-1} \)](image)

**Definition 4.3.** A path on a graph is a sequence of vertices \( x_1, x_2, \ldots, x_\ell \) such that, for every \( i \) such that \( 1 \leq i < \ell \), \( v_i \) and \( v_{i+1} \) are connected by an edge.

In other words, a path is a way of getting from some initial vertex \( x_1 \) to some final vertex \( x_\ell \) that always follows edges. If \( p \) is a path in a graph, we will denote the \( i \)th vertex it visits by \( p_i \).

**Definition 4.4.** Let \( V_{n,k} \) be the set of complex linear combinations of paths of length \( n \) starting at the leftmost vertex of the straight-line graph \( \Gamma_{k-1} \).

According to this definition, the set \( P_{n,k} \) of all paths of length \( n \) starting at the leftmost vertex of \( \Gamma_{k-1} \) forms a basis for \( V_{n,k} \). This means that \( V_{n,k} \) is a finite-dimensional vector space over \( \mathbb{C} \), although finding the dimension for a given \( n \) and \( k \) is a non-trivial task.

To define a representation \( \tau \) of \( TL_n \) in \( V_{n,k} \), it suffices to specify its values on the set of cupcaps \( E_i \), which generate \( TL_n \). Because any other element of the algebra can be built as a sum of products of the \( E_i \), the definition can be extended to the rest of the algebra using the homomorphism properties (4.1, 4.2). For example, we can define \( \tau(3A^4E_1 + E_2E_3) \) by the equation \( \tau(3A^4E_1 + E_2E_3) = 3A^4\tau(E_1) + \tau(E_2)\tau(E_3) \).

We will define \( \tau(E_i) \) by specifying its matrix elements with respect to the basis of paths in \( V_{n,k} \) one by one. To find the matrix element \( \tau_{pq}(E_i) \) in the row indexed by the path \( p \) and the column indexed by the path \( q \), we will use \( p \) and \( q \) to color the diagram representing \( E_i \) in the following way:

First, write the numbers \( p_i \) corresponding to the vertices visited by \( p \) in the gaps between strands along the top of the diagram, and the numbers corresponding to the vertices visited by \( q \) in the gaps along the bottom. For example, if \( p \) is the path going from vertex 1 \( \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 1 \) and \( q \) goes from 1 \( \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 1 \), we get the picture below:

![Diagram of \( \tau_{pq}(E_i) \)](image)

Now, pretending that each number represents a color, we can color in the remainder of the diagram based on the numbers along the top and bottom. This only works if there are no clashes, that is, if each of the connected regions of the diagram which are separated by the strands is labeled by only one number. If there are clashes of this sort, we declare the matrix element to be 0.
If there are no clashes, we look at the cap and cup-shaped regions of the diagram. Each of these is assigned a coefficient according to how the adjacent regions are colored, as per the rules below. The matrix element in question is the product of the coefficients corresponding to all the cap and cup-shaped regions in the diagram.

\[
\begin{array}{cccc}
\ell + 1 & \ell - 1 & \ell & \ell \\
\ell & \ell & \ell + 1 & \ell - 1 \\
\downarrow & \downarrow & \downarrow & \downarrow \\
a_\ell & b_\ell & c_\ell & d_\ell \\
\end{array}
\]

In order for this to define a representation of $T\ell_n$, the coefficients $a_\ell, b_\ell, c_\ell$ and $d_\ell$ must satisfy a certain set of constraints (see Exercise 2). For our purposes, we will take

\[
a_\ell = c_\ell = \frac{\sin \left( \frac{\pi (\ell + 1)}{\ell} \right)}{\sin \left( \frac{\pi \ell}{2} \right)} \quad \text{and} \quad b_\ell = d_\ell = \frac{\sin \left( \frac{\pi (\ell - 1)}{\ell} \right)}{\sin \left( \frac{\pi \ell}{2} \right)}.
\]

(4.3)

**Theorem 4.1.** If we let $A = ie^{-\frac{2\pi}{\ell}}$, then the map $\tau$ defined above is a representation of $T\ell_n$ in $V_{n,k}$.

**Proof.** Since $\tau$ is defined on the generators $E_i$ of $T\ell_n$ and extended to the rest of the algebra using the homomorphism properties (4.1, 4.2), we only need to check that the matrices $\tau(E_i)$ satisfy the relations (3.5). It will then follow that the function $\tau$ is an algebra homomorphism from $T\ell_n$ to $L(V_{n,k})$.

The element in row $p$ and column $q$ of the matrix product $\tau(E_i)\tau(E_j)$ is given by

\[
\sum_{r \in P_{n,k}} \tau_{p,r}(E_i)\tau_{r,q}(E_j).
\]

(4.4)

This amounts to summing over all the possible path-colorings of the middle of the diagram. However, the only nonzero terms in the sum will be those for which the path $r$ coloring the middle of the diagram is compatible with the colorings of the top and bottom of the diagram, since otherwise either $\tau_{p,r}(E_i)$ or $\tau_{r,q}(E_j)$ will be zero. It is straightforward to use this to show that, if $|i - j| \geq 2$, $\tau(E_i)\tau(E_j) = \tau(E_j)\tau(E_i)$ (see Exercise 3).

Consider the product $\tau(E_i)\tau(E_{i-1})\tau(E_i)$. The relevant portion of the product diagram is shown below.
It is clear that a pair of compatible colorings $p$ and $q$ of the top and bottom of the diagram uniquely determines the coloring at both intermediate points, so there will only be one nonzero term in the sum which gives the entry in row $p$ and column $q$ of the resulting matrix. Compared to the corresponding entry $\tau_{p,q}(E_i)$ in the matrix for $E_i$, this entry has four extra factors, which can be read off the diagram as the coefficients corresponding to the intermediate cups and caps. If $\ell \equiv p_{i-2} < p_{i-1}$, then the extra factor is

$$d_{\ell+1}^a \ell c_{\ell} b_{\ell+1} = \sqrt{\frac{\sin \left( \frac{\ell \pi}{k} \right) \sin \left( \frac{\ell (\ell+1) \pi}{k} \right) \sin \left( \frac{\ell q \pi}{k} \right) \sin \left( \frac{\ell (\ell+1) q \pi}{k} \right)}{\sin \left( \frac{\ell (\ell+1) \pi}{k} \right) \sin \left( \frac{\ell \pi}{k} \right) \sin \left( \frac{\ell q \pi}{k} \right) \sin \left( \frac{\ell (\ell+1) q \pi}{k} \right)}} = 1,$$

and if $\ell = p_{i-2} > p_{i-1}$, then it is

$$c_{\ell-1} b_{\ell} d_{\ell} a_{\ell-1} = \sqrt{\frac{\sin \left( \frac{\ell \pi}{k} \right) \sin \left( \frac{\ell (\ell-1) \pi}{k} \right) \sin \left( \frac{\ell q \pi}{k} \right) \sin \left( \frac{\ell (\ell-1) q \pi}{k} \right)}{\sin \left( \frac{\ell (\ell-1) \pi}{k} \right) \sin \left( \frac{\ell \pi}{k} \right) \sin \left( \frac{\ell q \pi}{k} \right) \sin \left( \frac{\ell (\ell-1) q \pi}{k} \right)}} = 1.$$

This means that the entry in row $p$ and column $q$ of the matrix $\tau(E_i)\tau(E_{i-1})\tau(E_i)$ is the same as the corresponding entry in $\tau(E_i)$ when the colorings corresponding to $p$ and $q$ are compatible. When they are not compatible, the entry on both sides is zero, so $\tau(E_i)\tau(E_{i-1})\tau(E_i) = \tau(E_i)$. The picture for $\tau(E_i)\tau(E_{i+1})\tau(E_i)$ is the same only flipped left-to-right, so an analogous argument applies.

Now consider the product $\tau(E_i)\tau(E_i) = \tau(E_i)^2$. Again, the relevant portion of the product diagram is shown below.
Given a pair of compatible colorings \( p \) and \( q \) of the top and bottom of the diagram, there are two compatible ways to color the middle – if \( \ell \equiv p - 1 \), then the central circle can be colored with either \( \ell + 1 \) or \( \ell - 1 \). This means that there are two terms in the sum (4.4) for the entry in row \( p \) and column \( q \) of the product matrix. By reading off the coefficients corresponding to the cap and cup portions of the central circle, we can see that the \((p,q)\)th entry in the product matrix is given by

\[
c_{\ell}a_{\ell}\tau_{p,q}(E_i) + d_{\ell}b_{\ell}\tau_{p,q}(E_i) = (a_{\ell}c_{\ell} + b_{\ell}d_{\ell})\tau_{p,q}(E_i).
\]

As in the previous case, this shows that the matrix entries corresponding to compatible pairs of colorings are equal, and those corresponding to incompatible pairs are all zero, so this shows that \( \tau(E_i)^2 = (a_{\ell}c_{\ell} + b_{\ell}d_{\ell})\tau(E_i) \). But

\[
a_{\ell}c_{\ell} + b_{\ell}d_{\ell} = \frac{\sin\left(\frac{\pi(\ell+1)}{k}\right)}{\sin\left(\frac{\pi\ell}{k}\right)} + \frac{\sin\left(\frac{\pi(\ell-1)}{k}\right)}{\sin\left(\frac{\pi\ell}{k}\right)}
\]

\[
= \frac{\sin\left(\frac{\pi\ell}{k}\right)\cos\left(\frac{\pi}{k}\right) + \cos\left(\frac{\pi\ell}{k}\right)\sin\left(\frac{\pi}{k}\right)}{\sin\left(\frac{\pi\ell}{k}\right)} + \frac{\sin\left(\frac{\pi\ell}{k}\right)\cos\left(\frac{\pi}{k}\right) - \cos\left(\frac{\pi\ell}{k}\right)\sin\left(\frac{\pi}{k}\right)}{\sin\left(\frac{\pi\ell}{k}\right)}
\]

\[
= 2\cos\left(\frac{\pi}{k}\right) = e^{-i\frac{\pi}{k}} + e^{i\frac{\pi}{k}} = -(ie^{-\frac{\pi}{k}})^2 - (ie^{\frac{\pi}{k}})^2
\]

\[
= -A^2 - A^{-2},
\]

so \( \tau(E_i)^2 = (-A^2 - A^{-2})\tau(E_i) \). Therefore the matrices \( \tau(E_i) \) satisfy the relations (3.5), so \( \tau \) is an algebra homomorphism from \( TL_n \) to \( L(V_{n,k}) \), and so \( \tau \) is a representation of \( TL_n \) in \( V_{n,k} \).

\[\square\]

**Exercises**

1. Let \( \rho \) be a representation of a group \( G \) in a vector space \( V \), and let \( e \) be the identity element of \( G \). Show that \( \rho(e) \) is the identity transformation on \( V \).

2. Show that, for the matrices \( \tau(E_i) \) to satisfy the relations (3.5), the coefficients \( a_{\ell}, b_{\ell}, c_{\ell} \) and \( d_{\ell} \) must satisfy \( a_{\ell}c_{\ell+1} = b_{\ell+1}c_{\ell} = 1 \) and \( a_{\ell}b_{\ell} + c_{\ell}d_{\ell} = -A^2 - A^{-2} \), and that any set of complex numbers \( a_{\ell}, b_{\ell}, c_{\ell} \) and \( d_{\ell} \) satisfying these equations defines a representation of \( TL_n \) in \( V_{n,k} \).

3. Complete the proof of Theorem 4.1 by arguing that \( \tau(E_i)\tau(E_j) = \tau(E_j)\tau(E_i) \) whenever \( |i - j| \geq 2 \).

4. Let \( \rho : B_n \to GL(V) \) be a representation of the braid group \( B_n \) in a finite-dimensional vector space \( V \), and define \( \chi : B_n \to \mathbb{C} \) by \( \chi(b) = \text{tr}(\rho(b)) \) for \( b \in B_n \). Show that if \( b_1 \) and \( b_2 \) are two \( n \)-strand braids representing the same knot, then \( \chi(b_1) = \chi(b_2) \).

5. Show that for \( k = 5 \), the dimension of the vector space \( V_{n,k} \) is the Fibonacci number \( F_{n+1} \), where \( F_1 = F_2 = 1 \) and \( F_n = F_{n-1} + F_{n-2} \) for \( n \geq 3 \).

6. The number of strings containing \( n \) pairs of correctly matched parentheses is given by the Catalan number

\[
C_n = \frac{1}{n + 1} \binom{2n}{n}
\]

Use this information to show that, for \( k \geq n + 2 \), the dimension of the vector space \( V_{n,k} \) is given by

\[
\dim V_{n,k} = \left\lfloor \frac{n}{2} \right\rfloor
\]

where \( \lfloor x \rfloor \) is the floor function.
4.2 An Alternative Description of the Path Model Representation

An alternate way to generate a matrix for the $TL_n$ elements $E_i$ is outlined in this section. Let $S = \{0, 1\}^n$. Note that we can also write $S = \{s_j | 0 \leq j \leq 2^n\}$, where $s_j$ is the binary representation of the integer $j$. For each $j$ between 1 and $2^n$, define a new sequence of numbers $x_{j1}, x_{j2}, x_{j3}, \ldots x_{jn}$ such that $x_{jk}' = -1$ if the $k$th binary digit of $j$ is 0 and 1 if the $k$th binary digit of $j$ is 1. Now let

$$L(j, m) = 1 + \sum_{k=1}^{m} x_{jk}'$$

With the understanding that we are trying to write a matrix for $E_i$, we will write $\ell = L(j, i)$. The entries of the matrix $\tau(E_i)$ are

$$\tau(E_i)_{pq} = \begin{cases} 
0 & \text{for any } a < n \text{ if } L(p, a), L(q, a) < 1 \\
& \text{or } L(p, a), L(q, a) > k \\
b^2_{\ell} & \text{for } s_p, x_p, x_{p+1} = 01 \\
& \text{and} \\
a^2_{\ell} & \text{for } s_q, x_q, x_{q+1} = 01 \\
& \text{and} \\
a_{\ell}b_{\ell} & \text{for } s_p, x_p, x_{p+1} = 10 \\
& \text{and} \\
0 & \text{for } s_q, x_q, x_{q+1} = 01 \text{ or } 10 \\
0 & \text{otherwise} 
\end{cases} \quad (4.5)$$

**Claim 4.1.** A matrix computed using these rules is the same as the one that arises from the Path Model Representation defined in the previous section.

**Example:** Using both this formulation and the path-model representation it can be shown that

$$\tau(E_1) = \begin{pmatrix} 
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{pmatrix}, \text{ and } \tau(E_2) = \begin{pmatrix} 
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{pmatrix}$$

**Exercises**

1. Compute $\tau(\sigma_2)$ and $\tau(\sigma_2^{-1})$ for $n = 2$ using both the path model representation and the formula given in equation 4.5. Are the same?
2. Change equation 4.5, such that the first condition about $L(p, a)$ and $L(q, a)$ is removed and compute $\tau(\sigma_2)$ and $\tau(\sigma_2^{-1})$ for $n = 2$. Is your answer the same as 1.2?
3. Write down all 4-bit strings and write $L(j, \alpha)$ for $\alpha < n$ for all these strings $s_j$. Also write the paths representing these strings. Which paths correspond to $L(j, \alpha) = 0$?

4.3 The Matrix for a Braid

Give a braid with $n$ strands, and $m$ crossings, our aim is to write a $2^n \times 2^n$ matrix representing that braid. Recall that using skein relations every braid word generator $\sigma_i$ and $\sigma_i^{-1}$ can be written using $TL_n$ elements:
\[\sigma_i = AE_i + A^{-1}E_0\]
\[\sigma_i^{-1} = AE_0 + A^{-1}E_i\]

**Example:** Recall that the knot figure-8 as a braid words is written as \(\sigma_2\sigma_1^{-1}\sigma_2\sigma_1^{-1}\). Using \(T_{L_\nu}\) elements this can be written as
\[\sigma_2\sigma_1^{-1}\sigma_2\sigma_1^{-1} = (AE_2 + A^{-1}E_0) \times (AE_0 + A^{-1}E_1) \times (AE_2 + A^{-1}E_0) \times (AE_0 + A^{-1}E_1)\]

Now, if \(W\) is the matrix representing the figure-8 knot, then
\[W = \tau(\sigma_2\sigma_1^{-1}\sigma_2\sigma_1^{-1}) = (\tau(E_2) + A^{-1}\tau(E_0)) \times (\tau(E_0) + A^{-1}\tau(E_1)) \times (\tau(E_2) + A^{-1}\tau(E_0)) \times (\tau(E_0) + A^{-1}\tau(E_1))\]

Using the information about \(\tau(E_1)\) and \(\tau(E_2)\) from above, we get
\[
W = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
A^4 + ((a_1b_2)^2 + 2a_2^2)A^2 + 0
\]

4.4 **The Jones Polynomial in the Matrix Formulation**

**Theorem 4.2.** If a matrix \(W\) appropriately represents the knot \(K\), then given a complex number \(a\), there exists a transformation \(f\) such that the complex number \(f(W)\) is the Markov trace, with \(A = a\). In particular, \(f(W) = \frac{1}{N} \sum_{j=1}^{2^\nu} \lambda_{\ell_j} W_{jj}\), where \(\ell_j = \begin{cases} \mathcal{L}(j,n) & \text{for all } n \leq j \leq \mathcal{L}(j,a) < k \\ 0 & \text{otherwise} \end{cases}\) and \(N = \sum_{j=1}^{2^\nu} \lambda_{\ell_j}\) is the normalization factor.

(See Lemma 2.17 (page 409), and Claim 3.11 (page 417) in Aharonov et al.’s “A Polynomial Quantum Algorithm for Approximating the Jones Polynomial” for the proof of this theorem)

\(f(W)\) can be thought of as a weighted average of the diagonal elements of \(W\). Also note that \(f(W)\) depends on the complex number \(a\). In particular, when the number of colors is \(k\) then \(A = i\varepsilon_k\). This is not an intuitive result, but is fundamentally important for using the quantum algorithm to compute the Jones Polynomial corresponding to a braid.

**Example:** Consider the trivial case i.e. \(E_0\). \(\tau(E_0) = 1\) and \(\tau(E_0) = I\).

\[f(\tau(E_0)) = \frac{1}{N} \sum_{j=1}^{2^\nu} \lambda_{\ell_j} \tau(E_0) = \frac{1}{2^\nu} \sum_{j=1}^{2^\nu} \lambda_{\ell_j} \frac{1}{1} = 1\]

**Example:** Consider the braid \(\sigma_1\) with \(k = 5\). The Markov Trace for this braid \(\tau(\sigma_1) = \tau(E_1) + A^{-1}\tau(E_0)\). Note that \(\tau(E_1) = (A^{-2} - A^2)^{-1}\), as shown below, and \(\tau(E_0) = 1\). So, \(\tau(\sigma_1)A(-A^{-2} - A^2)^{-1}A^{-1}\). Now consider the matrix formulation, using either the path model representation or equation 4.5 we can find that

\[
\tau(E_1) = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & a_1^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]
and therefore,

\[ \tau(\sigma_1) = \begin{pmatrix} A^{-1} & 0 & 0 & 0 \\ 0 & A^{-1} & 0 & 0 \\ 0 & 0 & A(a_1^2) + A^{-1} & 0 \\ 0 & 0 & 0 & A^{-1} \end{pmatrix}. \]

We want to show that \( f(\tau(\sigma_1)) = \text{tr}(\tau(\sigma_1)). \) Here \( N = \lambda_1 + \lambda_2 \)

\[ f(\tau(\sigma_1)) = \frac{1}{N} \sum_{j=1}^{2n} \lambda_j \tau(\sigma_1)_{jj} = \frac{\lambda_1 (Aa_1^2 + A^{-1}) + \lambda_2 A^{-1}}{\lambda_1 + \lambda_2} = A \left( \frac{\lambda_1 a_1^2}{\lambda_1 + \lambda_2} \right) + A^{-1} \]

Setting \( A = i e^{\frac{i\pi}{n}} \), you can show that \((-A^{-2} - A^2)^{-1} = \frac{(\lambda_1 a_1^2)}{\lambda_1 + \lambda_2} \), and hence

\[ \text{tr}(\sigma_1) = \text{tr}(\sigma_1) A(-A^{-2} - A^2)^{-1} + A^{-1} = A \left( \frac{\lambda_1 a_1^2}{\lambda_1 + \lambda_2} \right) + A^{-1} = f(\tau(\sigma_1)) \]

Exercises

1. Compute Markov Trace for braid word \( \sigma_1 \sigma_2 \) first by computing the Markov Trace for \( TL_n \) elements and secondly using the weighted trace method described above.

2. Compute the Markov Trace for \( \sigma_1 \sigma_1 \sigma_1 \) using the weighted trace method. Compare this to the Jones Polynomial of this same braid, found using whatever method you prefer.
5 Linear Algebra

Definition 5.1. Given a vector space $V$ over $\mathbb{C}$ an inner product $\langle , \rangle$ on $V$ is a function $\langle , \rangle : V \times V \rightarrow \mathbb{C}$ such that

- $\langle \vec{v}, \vec{w} \rangle = \langle \vec{w}, \vec{v} \rangle$ for all $\vec{v}, \vec{w} \in V$.
- $\langle \vec{v}_1 + \vec{v}_2, \vec{w} \rangle = \langle \vec{v}_1, \vec{w} \rangle + \langle \vec{v}_2, \vec{w} \rangle$, for all $\vec{v}_1, \vec{v}_2, \vec{w} \in V$.
- $\langle a\vec{v}, \vec{w} \rangle = a \langle \vec{v}, \vec{w} \rangle$ for all $\vec{v}, \vec{w} \in V$ and for all $a \in \mathbb{C}$.
- If $\vec{v} \neq \overrightarrow{0}$ then $\langle \vec{v}, \vec{v} \rangle > 0$.

Source: [6].

Definition 5.2. Let $V$ be a vector space over $\mathbb{C}$. The adjoint of a linear map $A : V \rightarrow V$ is the linear map $A^\dagger$ whose matrix with respect to any orthonormal basis is given by the conjugate transpose of $A$ i.e. $A^\dagger = \overline{A^T}$.

Definition 5.3. This square matrix $A$, with complex entries, is called Hermitian if $A^\dagger = A$.

Definition 5.4. Let $V$ be a vector space over $\mathbb{C}$ equipped with an inner product $\langle , \rangle$. A linear map $A : V \rightarrow V$ is unitary if $A^\dagger A = I$.

Unitary maps are special because they preserve the length of vectors that they act upon.

Definition 5.5. Let $V$ and $W$ be vector space over $\mathbb{C}$ of $n$ and $m$ dimensions respectively. The basis vectors these two spaces can be represented as $B_V = \{ \vec{e}_1, \vec{e}_2, \vec{e}_3, \ldots, \vec{e}_n \}$ and $B_W = \{ \vec{f}_1, \vec{f}_2, \vec{f}_3, \ldots, \vec{f}_m \}$. The direct sum of these two vector spaces if the vector space spanned by the set $B_V \cup B_W$.

If $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$ and $\vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$ then $\vec{v} \oplus \vec{w} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$.

Likewise, for matrices $A : V \rightarrow V$ and $B : W \rightarrow W$,

$$A \oplus B = \begin{pmatrix} A & 0_{n \times m} \\ 0_{m \times n} & B \end{pmatrix}.$$ 

Source: [3].

Definition 5.6. Let $V$ and $W$ be vector spaces over $\mathbb{C}$ of $n$ and $m$ dimensions respectively, and suppose that $B_V = \{ \vec{e}_1, \vec{e}_2, \vec{e}_3, \ldots, \vec{e}_n \}$ and $B_W = \{ \vec{f}_1, \vec{f}_2, \vec{f}_3, \ldots, \vec{f}_m \}$ are bases for $V$ and $W$, respectively. The tensor product $V \otimes W$ is the vector space over $\mathbb{C}$ with basis $B_{V \otimes W} = \{ \vec{e}_i \otimes \vec{f}_j | \vec{e}_i \in V, \vec{f}_j \in W \}$. 

Source: [3].

21
If \( \vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \) and \( \vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} \) then \( \vec{v} \otimes \vec{w} = \begin{pmatrix} v_1 w_1 \\ v_1 w_2 \\ \vdots \\ v_1 w_m \\ v_2 w_1 \\ v_2 w_2 \\ \vdots \\ v_2 w_m \\ \vdots \\ v_n w_1 \\ v_n w_2 \\ \vdots \\ v_n w_m \end{pmatrix} \).

Likewise, for matrices \( A : V \to V \) such \( W = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} \) and \( B : W \to W \),

\[
A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & a_{13}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & a_{23}B & \cdots & a_{2n}B \\ a_{31}B & a_{32}B & a_{33}B & \cdots & a_{3n}B \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & a_{n2}B & a_{n3}B & \cdots & a_{nn}B \end{pmatrix},
\]

where \( A \otimes B \) is a \( nm \times nm \) matrix. It is also worth noting that \( (A \otimes B) \cdot (\vec{v} \otimes \vec{w}) = A \vec{v} \otimes B \vec{w} \). Source: [3].

**Exercises**

1. Show that \( \text{tr}(A \otimes B) = \text{tr}(A) + \text{tr}(B) \).
2. Show that \( \text{tr}(A \otimes B) = \text{tr}(A) \text{tr}(B) \).
3. If \( V \) is a 2-dimensional vector space and \( W \) is a 3-dimensional vector space, write the basis vectors for \( V \otimes W \) and \( V \oplus W \), using the standard basis for \( V \) and \( W \).
4. Show that \( (A_1 \otimes B_1) \cdot (A_2 \otimes B_2) = (A_1 A_2) \otimes (B_1 B_2) \), where \( A_i \) and \( B_i \) are 2 \( \times \) 2 and 3 \( \times \) 3 matrices respectively.
5. Show that \( (A \otimes B) \cdot (\vec{v} \otimes \vec{w}) = (A \vec{v}) \otimes (B \vec{w}) \).
6. Suppose that \( N \) is a matrix that commutes with its adjoint, so that \( N^\dagger N = NN^\dagger \), and there is a positive integer \( m \) such that \( N^m = I \), the identity matrix. Show that \( N \) is unitary. (Hint: Put \( N^\dagger N \) in upper triangular form. What are the diagonal entries?)
6 The Circuit Model of Quantum Computation

(See Preskill [8] Section 5.3 for a more in-depth description of the quantum circuit model.)

6.1 Overview: What Is Quantum Computing?

Quantum mechanics: A well-supported theory of physics consisting of mathematical descriptions for the behavior of subatomic particles. Quantum computing came from the idea that the complex (and often counter-intuitive) mathematical properties of quantum particles could be used to perform computation in a new way.

How do we understand the classical-quantum difference? A quantum system of \( n \) particles is described by a Hilbert space which has \( 2^n \) dimensions. Because of this, no classical computer seems to be able to efficiently simulate quantum systems. Nevertheless, the quantum system is able to “simulate” itself efficiently just fine.

Practical Relevance: It has been shown that we can design quantum systems that factor integers asymptotically faster than is possible using any known algorithm for a classical computer. Because of this, it is believed that the construction of a quantum computer will break most of the security (in particular, RSA) in use on the Internet today.

Standard, computationally equivalent models of quantum computing include:

- Quantum Turing machine
- Adiabatic quantum computer
- Topological quantum computer
- Quantum circuit

For our purposes we focus on a specific model: the quantum circuit model.

1. This model is a theoretical abstraction of quantum phenomena which is useful for describing quantum algorithms.
2. This model is computationally equivalent to other standard models of quantum computing.

6.2 Qubits and Quantum Gates

(See Section I.B, “Introduction to linear optics quantum sampling” from Dowling et al.’s An Introduction to Boson-Sampling (page 3) [4]).

Recall that classical bits can take on one or two values, generally represented by 0 and 1. \( n \)-bit strings are strings of zeroes and ones – for instance, 001 is 3-bit string. Given a positive integer \( n \), there are \( 2^n \) possible \( n \)-bit strings. Logical operations on classical bits are done using logic gates.

**Definition 6.1.** A classical logic gate is a function \( f : \{0,1\}^n \rightarrow \{0,1\}^m \), that is, an function which acts on \( n \)-bit strings and produces \( m \)-bit strings.

Simple examples of classical logic gates include the logical operations AND, OR, and NOT, where 0 is interpreted as “False” and 1 is interpreted as “True.” AND and OR gates operate on 2-bit strings and produce single-bit outputs, whereas NOT gates operate on single bits and produce single bits. Tables of values for the AND and OR gates are shown below, demonstrating that classical logic gates are not necessarily invertible (bijective).

**Example 6.1.** Tables of values for the functions AND : \( \{0,1\}^2 \rightarrow \{0,1\} \) and OR : \( \{0,1\}^2 \rightarrow \{0,1\} \).

Qubits are different from bits in that a qubit can exist in one of infinitely many states. \( n \)-qubit states live in a vector space generated by the basis \( B = \{ |v_s\rangle \ s \in S \} \), where \( S \) is the set of \( n \)-bit strings. The basis elements are generally ordered as an ascending sequence of binary numbers.
\[ \begin{array}{cccc}
\text{Input} & 00 & 01 & 11 \\
\text{AND} & 0 & 0 & 0 & 1 \\
\text{OR} & 0 & 1 & 1 & 1 \\
\end{array} \]

Table 1: Logic Gates AND and OR

**Definition 6.2.** Let \( \mathcal{H}_n \) be the vector space spanned by the basis \( B \) defined above. The **standard inner product** \( \langle \cdot, \cdot \rangle \) on \( \mathcal{H}_n \) is the inner product on \( \mathcal{H}_n \) defined by taking \( B \) to be orthonormal. That is,

\[
\sum_{s \in S} a_s \tilde{v}_s, \sum_{s \in S} b_s \tilde{v}_s = \sum_{s \in S} \tilde{a}_s b_s.
\]

**Definition 6.3.** An **n-qubit state** is a vector \( \tilde{v} \in \mathcal{H}_n \) such that

\[
\langle \tilde{v}, \tilde{v} \rangle = 1.
\]

**Example 6.2.** There are 8 classical 3-bit strings, as listed below:

\[ S = \{000, 001, 010, 011, 100, 101, 110, 111\} \]

The vector space in which 3-qubit states live hence has 8 basis vectors, one for each of the bit-strings above. The following is an example of a state of a 3-qubit system:

\[
\frac{1}{5} \tilde{v}_{000} + \frac{2}{5} \tilde{v}_{001} + i \frac{2}{5} \tilde{v}_{010} + \frac{3}{5} \tilde{v}_{011} - \frac{1}{5} \tilde{v}_{100} - \frac{2}{5} \tilde{v}_{101} - i \frac{1}{5} \tilde{v}_{110} + \frac{2}{5} \tilde{v}_{111}.
\]

Just as classical logic gates take \( n \)-bit strings to \( m \)-bit strings, quantum logic gates take one \( n \)-qubit state to another. In contrast to classical logic gates, which need not be reversible, all quantum logic gates are reversible.

**Definition 6.4.** A quantum logic gate \( T \) is a unitary linear transformation \( T : \mathcal{H}_n \rightarrow \mathcal{H}_n \). Such a transformation can be represented by a \( 2^n \times 2^n \) unitary matrix.

An example of a two-qubit quantum logic gate is the CNOT gate, which executes NOT on the second qubit if the first bit is 1 and does nothing if the first qubit is 0. This gate is represented by the matrix below.

\[
\text{CNOT} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}.
\]

**6.2.1 Universal Gates**

(See Universal Quantum Gates by Jean-Luc Brylinski and Ranee Brylinski [5] for a proof about existence of universal quantum gates.)

**Definition 6.5.** A set of gates \( \mathcal{U} \) is called universal if, for each \( n \geq 2 \), every \( n \)-qubit gate \( T \) can be approximated, with arbitrary accuracy, by a quantum circuit made up of quantum logic gates in \( \mathcal{U} \).

One commonly used set of gates that is universal is:

\[
\mathcal{U} = \{ \text{CNOT}, H, \alpha_x, \alpha_y, \alpha_z, \text{PHASE}, [\pi/8] \},
\]

where \( \text{CNOT} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix}
1 & 1 \\
1 & -1
\end{pmatrix}, \alpha_x = \begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}, \alpha_y = \begin{pmatrix}
0 & -i \\
i & 0
\end{pmatrix}, \alpha_z = \begin{pmatrix}
1 & 0 \\
0 & -1
\end{pmatrix}, \text{PHASE} = \begin{pmatrix}
1 & 0 \\
i & 0
\end{pmatrix}, [\pi/8] = \begin{pmatrix}
1 & 0 \\
0 & e^{i\pi/4}
\end{pmatrix} \text{[4].}
\]
Example: Consider the $4 \times 4$ unitary matrix $\begin{pmatrix} 0 & 0 & 0 & i \\ 0 & 0 & -i & 0 \\ 0 & -i & 0 & 1 \\ i & 0 & -i & 0 \end{pmatrix}$. This matrix can be written using Pauli matrices $\sigma_x$, $\sigma_y$ and $\sigma_z$ in the following form.

$$\begin{pmatrix} 0 & 0 & 0 & i \\ 0 & 0 & -i & 0 \\ 0 & -i & 0 & 1 \\ i & 0 & -i & 0 \end{pmatrix} = (\sigma_y \otimes I) \cdot (I \otimes \sigma_y) \cdot (\sigma_z \otimes I)$$

Exercises

1. Show that the vector $\tilde{u} = \frac{1}{\sqrt{5}} \hat{v}_{000} + \frac{2}{\sqrt{5}} \hat{v}_{001} + \frac{2}{\sqrt{5}} \hat{v}_{010} + \frac{1}{\sqrt{5}} \hat{v}_{011} - \frac{2}{\sqrt{5}} \hat{v}_{100} - \frac{1}{\sqrt{5}} \hat{v}_{101} + \frac{2}{\sqrt{5}} \hat{v}_{110} + \frac{2}{\sqrt{5}} \hat{v}_{111}$ is normalized i.e the length of this vector is 1.

2. Apply CNOT gate to vectors $\tilde{v}_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$, $\tilde{v}_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$, $\tilde{v}_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$, and $\tilde{v}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ and report the effect of applying this gate on these vectors.

3. Write the unitary matrix representing the logical function NOT i.e. NOT $\tilde{v}_0 = \tilde{v}_1$ and NOT $\tilde{v}_1 = \tilde{v}_0$.

4. Verify that $\begin{pmatrix} 0 & 0 & 0 & i \\ 0 & 0 & -i & 0 \\ 0 & -i & 0 & 1 \\ i & 0 & -i & 0 \end{pmatrix} = (\sigma_x \otimes I) \cdot (I \otimes \sigma_y) \cdot (\sigma_z \otimes I)$.

5. Write the following matrix using the gates from $\mathcal{U}$: $\begin{pmatrix} i & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & i \end{pmatrix}$ (Hint: You will only need to use the Pauli matrices.)

6.3 Computation


A computation under the quantum model can be thought of as the construction of a single quantum circuit.

Figure 10: An example of a quantum circuit with 6 qubits.

There are three steps involved in a computation:

1. Initialization of an $n$-qubit register $\psi(x_1, \ldots, x_6)$ in the example)
2. Application of universal gates (the four rectangles in the example)

3. Measurement of the first qubit (y in the example)

The physical implementation of this process is very difficult, but for our purposes we are concerned with the theoretical mathematical description. In linear algebraic terms, the steps can be understood as follows:

1. Choice of a normalized complex vector \( \psi \in \mathbb{C}^{2^n} \)
2. Multiplication of \( \psi \) with a \( 2^n \times 2^n \) unitary matrix \( U \in \mathbb{C}^{2^n} \times \mathbb{C}^{2^n} \) to get the vector \( U \psi \)
3. Measurement of a single bit \( y \in \{0, 1\} \), giving an approximation of \( \psi^\dagger U \psi \)

**Example:** The circuit diagram for the sequence of logic gates given by \( (\sigma_x \otimes I) \cdot (I \otimes \sigma_y) \cdot (\sigma_z \otimes I) \) is as follows:

This is a two qubit system, where \( \sigma_x \) first acts on the first qubit, then \( \sigma_y \) acts on the second qubit and finally \( \sigma_z \) acts on the first qubit.

**Example:** Consider a slightly more complicated circuit given below.

This circuit configuration can be written as \( (\sigma_x \otimes I \otimes H) \cdot (H \otimes \sigma_y \otimes I) \cdot (I \otimes \text{CNOT}) \)

**Exercises**

1. Write the equation corresponding to the following quantum circuit and write the unitary matrix \( U \) corresponding to this configuration.
2. Write down the unitary matrix $U$ given by $(\sigma_x \otimes I \otimes H) \cdot (H \otimes I \otimes I) \cdot (I \otimes \text{CNOT})$.

3. Draw the circuit representing the following sequence of gates, $U = (H \otimes I) \cdot (I \otimes \text{CNOT}) \cdot (H \otimes I)$ and compute the unitary matrix $U$ that represents this circuit.

4. Apply $U$ from 3. to the vector \( \vec{u} = \frac{1}{5} \vec{v}_{000} + \frac{2}{5} \vec{v}_{001} + \frac{1}{5} \vec{v}_{010} + \frac{2}{5} \vec{v}_{011} - \frac{1}{5} \vec{v}_{100} - \frac{2}{5} \vec{v}_{101} + \frac{1}{5} \vec{v}_{110} + \frac{2}{5} \vec{v}_{111} \).

Convince yourself that $U$ is unitary, by showing that $U \vec{u}$ is normalized.
7 A Quantum Algorithm for the Jones Polynomial

Recall: Using the path-model representation (or the formula in 4.5), the matrices of $T \ell_n$ elements can be computed and hence we can write the matrix from the braid words for any knot $K$. Moreover, given a matrix $W$ that appropriately represents a braid $K$, the Markov trace for $K$ is equal to $\frac{1}{N} \sum_{j=1}^{2^n} \lambda_j W_{jj}$, where $N \sum_{j=1}^{2^n} \lambda_j$ i.e. a weighted average of the diagonal elements of $W$. The diagonal elements for $W$ can be written as $W_{ij} = \bar{v}_i^\dagger W \bar{v}_j$. Even though $W$ is a $2^n \times 2^n$ matrix we are only interested in the diagonal elements.

\[
W = \begin{pmatrix}
  w_{11} & w_{12} & w_{13} & \ldots & w_{12n} \\
  w_{21} & w_{22} & w_{23} & \ldots & w_{22n} \\
  w_{31} & w_{32} & w_{33} & \ldots & w_{32n} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  w_{2n1} & w_{2n2} & w_{2n3} & \ldots & w_{2n2n}
\end{pmatrix}
\]

7.1 Hadamard Test

(See Kauffman, 2010, “Topological Quantum Information, Khovanov Homology and the Jones Polynomial”, pages 18-19 for a discussion about Hadamard test. Take note that this paper uses bra-ket notation.)

Goal: Extract information about the diagonal elements of a unitary matrix $W$ given the corresponding quantum logic gate.

An intuitive way to think about the Hadamard test, is to consider it a "black-box" where given a gate and a bit string corresponding to some diagonal entry in the matrix is given, the "black-box" gives the diagonal entry. In reality, the Hadamard test is more complicated than that.

\[
\begin{array}{c}
W \xrightarrow{\text{HADAMARD TEST}} W_{jj} \\
\bar{v}_j \xrightarrow{\text{HADAMARD TEST}} W_{jj}
\end{array}
\]

Hadamard Test gives us the diagonal entries of $W$

Before we look at the specifics of the Hadamard test, it is useful to consider the effect of Hadamard gate $H$, on individual qubits. Consider a single qubit system generated by the Hilbert space with basis $B = \{\bar{v}_0, \bar{v}_1\} = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$

\[
H \bar{v}_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (\bar{v}_0 + \bar{v}_1)
\]

\[
H \bar{v}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} (\bar{v}_0 - \bar{v}_1)
\]

Recall that Hadamard gate is one of the gates in the universal gate set $U$ (refer to section 7.2.1). To perform the Hadamard test, the gate $W$ must be sandwiched between two Hadamard gates $H$ as shown in the circuit diagram below.
Figure 11: Circuit diagram for the Hadamard test

The unitary gate $H$ is represented by the matrix $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. $\oplus$ represents a switch, which controls when the matrix $W$ will be applied. In matrix form, the combination of the switch and the gate $W$ is written as $I \oplus W$. Note that $\oplus$ in the equation represents a direct sum, while the symbol $\oplus$ in the circuit represents a switch, they are not necessarily the same thing. Moreover, the matrix for $I \oplus W = \begin{pmatrix} I & 0 \\ 0 & W \end{pmatrix}$. written in block form. Here the switch acts on the first (control) qubit and $W$ acts on the $2^n$ (normal qubits). As shown below, the switch controls the action of $W$ on the normal qubits, depending on whether the the first qubit is $\tilde{v}_0$ or $\tilde{v}_1$.

$$
(I \oplus W) \cdot (\tilde{v}_0 \otimes \tilde{\psi}) = \begin{pmatrix} I & 0 \\ 0 & W \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \tilde{\psi} = \begin{pmatrix} I & 0 \\ 0 & W \end{pmatrix} \cdot \begin{pmatrix} \tilde{\psi} \\ 0 \end{pmatrix} = (\tilde{v}_0 \otimes \tilde{\psi})
$$

$$
(I \oplus W) \cdot (\tilde{v}_1 \otimes \tilde{\psi}) = \begin{pmatrix} I & 0 \\ 0 & W \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \tilde{\psi} = \begin{pmatrix} I & 0 \\ 0 & W \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \tilde{\psi} \end{pmatrix} = (\tilde{v}_0 \otimes \tilde{W} \tilde{\psi})
$$

Having applied the Hadamard test, the first qubit is measured via the Hermitian matrix by $Q = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$. To find the real component of the diagonal elements of $W$, we will set the control qubit to be $\tilde{v}_0$ and the Hadamard test, as represented in the circuit diagram above can be written using the following equation:

$$
\tilde{\psi} = (Q \otimes I) \cdot (H \otimes I) \cdot (I \otimes W) \cdot (H \otimes I) \cdot (\tilde{v}_0 \otimes \tilde{\psi})
$$

$$
= (\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \otimes I) \cdot (\begin{pmatrix} 1 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 & -1 \end{pmatrix} \otimes I) \cdot (I \otimes W) \cdot (H \otimes I) \cdot (\tilde{v}_0 \otimes \tilde{\psi})
$$

$$
= \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} I \\ 0 \\ W \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} \tilde{\psi} \\ 0 \end{pmatrix}
$$

$$
= \frac{1}{2} \begin{pmatrix} (I + W)\tilde{\psi} \\ 0 \end{pmatrix}
$$

$$
\Rightarrow \tilde{\psi} = \frac{1}{2} (1 + \text{Re}(\tilde{\psi}^\dagger W \tilde{\psi}))
$$

Therefore, we get the real part of the chosen diagonal element using the Hadamard test. A similar calculation can be done to get the imaginary part of the diagonal elements. The only difference would be to set the control qubit to be $\tilde{v}_1$ instead of $\tilde{v}_0$ and $\tilde{v}_1$.

**Exercises**

1. Write a switch, similar to $\oplus$ in figure 11, where $W$ gets applied when the first qubit is $\tilde{v}_0$.

2. What would happen if the control qubit was set to be $\tilde{v}_1$ instead of $\tilde{v}_0$? Do we get any information about $W$ using this construct?
3. Show that we can find \( Im(W_{jj}) \) using the Hadamard test when the control qubit is \( \frac{1+i}{2} v_0^* + \frac{1-i}{2} v_1^* \).

4. Let \( W = \sigma_1 \). Perform the computation as shown in the section above, to show that you can in fact find the diagonal elements of \( W \) using the Hadamard test.

7.2 Quantum Algorithm

(See page 416 in [2] for the description of the algorithm given in the original paper.)

**Input:** A braid word \( B \) of \( n \) strands and \( m \) crossings, and an integer \( k \).

**Output:** The value of the Jones polynomial of the trace closure of \( B \) at the root of unity \( A = i e^{-\frac{\pi i}{2}} \).

1. Classically, calculate the writhe, \( w \), of the trace closure of \( B \).

2. Classically, pick a random path \( p \in P_{n,k} \) with probability \( \Pr(p) \propto \lambda_{\ell} \) where \( \ell \) is the index of the ending point of \( p \).

3. Repeat for \( j = 1 \) to \( \text{poly}(n,m,k) \) the following quantum circuit computation in \( n \) qubits:
   - Initialize the input register with the basis vector, \( \psi \), described by the bitstring form of \( p \).
   - Apply quantum gates which encode the matrix representation, \( W \), of \( B \).
   - Perform the Hadamard test to output \( x_j \) with expected value \( \text{Re}\langle\psi|W|\psi\rangle \)

4. Do the same, but with a variant of the Hadamard test, to output a sequence of \( y_j \) with expected value \( \text{Im}\langle\psi|W|\psi\rangle \)

5. Let \( \mu \) be the average over all \( x_j + iy_j \). Output \( (-A)^{-3w}(-A^2 - A^{-2})^{n-1} \mu \) where \( A = i e^{-\frac{\pi i}{2}} \).
8 Analysis of Algorithmic Complexity

8.1 What is Polynomial Time Complexity and Why Do We Care?
(See pages 20–21 (section 1.7) in [8] and the last three paragraphs of the first subsection of 5.3 (pages 25–26, preceding 5.3.1) in [9] for discussions of polynomial time and its relationship to quantum computing.)

Easy and Hard—The Aim of Complexity Theory: Computational complexity theory aims to describe the inherent difficulty of computational problems in terms of how the amount of resources required to solve a problem scales with the size \( n \) of the input, in bits. More specifically, we could say that our goal is to have a hardware-independent way of describing problems as either “easy” or “hard.”

Polynomial Time: It turns out that it is both useful from a practical standpoint and interesting from a theoretical standpoint to use polynomial time complexity as the cutoff for “easy” problems. We say a problem \( A \) is of polynomial time complexity or that it is solvable in polynomial time if there exists an algorithm that solves \( A \) and that runs on a Turing machine in a number of steps that is upper-bounded by some polynomial function of the input size \( n \). We call the set of such problems \( P \). We often call these problems “efficiently solvable” and call an algorithm that runs in polynomial time “efficient.”

\[ P = \{ \text{problems solvable in polynomial time on a Turing machine} \} = \{ \text{"efficiently" solvable problems on a classical computer} \} \]

Quantum Polynomial Time: Complexity theory began under the hypothesis, sometimes called the strong Church–Turing thesis, that any reasonable method of physical computation can be efficiently simulated by any other. This principle is threatened by discoveries in quantum computing which suggest that some problems are efficiently solvable on a quantum computer but not a classical computer. We referred to one such problem in 7.1, namely, the problem of simulating a quantum system.

The complexity class \( BPP \) contains those problems solvable in polynomial time on a probabilistic Turing machine with low, bounded error. \( BQP \) contains those problems solvable in polynomial time on a quantum Turing machine with low, bounded error. To restate what was said in the preceding paragraph, it is conjectured that \( BPP \neq BQP \).

\[ BQP = \{ \text{"efficiently" solvable problems on a quantum computer} \} \]

Theorem 8.1. \( P \subseteq BPP \subseteq BQP \)

Conjecture 8.1. \( BPP \subseteq BQP \)

8.2 Circuit Complexity
(See section 5.1 (especially pages 7–8 in 5.1.3) in [9] for a discussion of classical and quantum circuit complexity.)

Algorithmic complexity under the quantum circuit model can be understood as analogous to complexity under the classical circuit model.

Classical Circuits: A classical circuit can be described as a collection of logical gates (e.g. NOT, AND, OR) connected in a directed acyclic graph, with some number of bits of input and output. In essence, the circuit encodes a function \( f : \{0,1\}^n \rightarrow \{0,1\}^m \). A computation, then, consists in the application of this circuit to a specified string of input bits. We defined \( P \) above in terms of the Turing machine model of computation, but we can also state it, equivalently, under the circuit model.

\[ P = \{ \text{problems solved by polynomial-size uniform circuit families} \} \]

In other words: problems for which we can design, in polynomial time, a circuit with a polynomial number of gates which computes the answer from the input.
Theorem 8.2. A problem is in BQP iff it is solved by a polynomial-size quantum circuit family.

(One direction of this biconditional can be found on p. 53 in [9] and p. 354 (Theorem 2) in [10]. We have not been able to find an explicit statement of the other direction, but it is assumed in the proof of Claim 3.6 on page 416 in [2]. It also seems to be implicit in [9], but it is not explicitly stated.)

Theorem 8.3. A unitary transformation which operates on a number of qubits logarithmic in the input size can be implemented in a number of universal gates polynomial in the input size.

(Referred to on p. 415 in [2]. Can be deduced from Theorem 1 on p. 353 in [10])

8.3 Analysis of the Quantum Algorithm for the Jones Polynomial

We consider the time complexity of each of the steps described in 8.2 above.

1. This is assumed to be easy.
2. This is efficient using the algorithm described in the proof of Claim 3.7 on p. 416 in [2].
3. Per Theorem 9.3 above, a single computation on this circuit takes polynomial time. A polynomial number of repetitions therefore takes polynomial time.
4. This has the same running time as step 3.
5. This is easy on a classical computer. In fact, assuming we take the sum of our outputs as we go in steps 3 and 4, this calculation takes constant time.
9 Conclusion

Why calculate the Jones polynomial?

- **Knot Invariant:** As mentioned in 2.2, the Jones polynomial is a powerful link invariant. Being able to calculate it efficiently is therefore useful for the purpose of identifying and differentiating knots and links.

- **Generalizations:** The Jones polynomial is related to other mathematical structures such as the colored Jones polynomial, the Potts model, and the Tutte polynomial. The algorithm presented here has been generalized to these other structures.

Why use a quantum algorithm?

- **Quantum Speedup:** The problem of approximating the Jones polynomial for the plat closure of a braid is BQP-complete. It is therefore unlikely that it can be efficiently solved on a non-quantum computer.

- **Deep Connection to Quantum** From previous work on topological quantum field theory, it was already known that the calculation of the Jones polynomial was well suited to quantum computation, and it was known that an algorithm like the one presented here must exist.

- **New Kind of Quantum Algorithm:** Previous quantum algorithms fall into a few categories: e.g. number theoretic, often based on the Fast Fourier Transform. This is the first application of quantum computing to a combinatorial problem.

Why does the algorithm work the way it does?

- **Unitary Representation:** Quantum computation is well suited to work with unitary matrices. We were able to represent a knot as a unitary matrix, and prove that the Jones polynomial was equivalent to a trace on that matrix.

Where does the quantum speedup come from?

- A quantum system is able to model an exponentially large unitary transformation without representing that transformation explicitly, in the way we would in a classical computation. In this algorithm, we apply a series of gates, each of which is described by a small \((4 \times 4)\) matrix, but has an effect on the state of the quantum system which is modeled by an exponentially large matrix.
References


