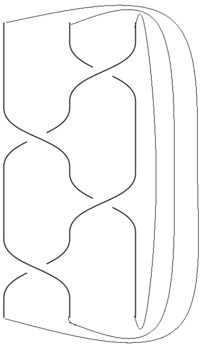
Quantum algorithm for Jones polynomials using Q#

I am a graduate student in Japan studying knot theory. My interest in quantum computer comes from learning about some quantum algorithms related to topology in math (e.g. knot theory, 3-manifolds, topological data analysis, etc.), and ideas of braids used in hardware. In this post, we will see the quantum algorithm for Jones polynomials using Q# for simple examples.

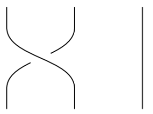
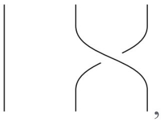
Goal

The goal of this post is to compute the approximate value of the Jones polynomial of the figure-eight knot at the imaginary unit $i = \sqrt{-1}$ and the 5th root of unity $e^{\frac{2\pi i}{5}}$. From the viewpoint of knot theory, the Jones polynomial is one of the tools to classify knots. Since the relation with the quantum field theory was found, Jones polynomials are now intriguing object to study. In particular, values at the root of unity $e^{\frac{2\pi i}{n}}$ $(n = 1, 2, 3, \cdots)$ relates to the long-time conjecture called Volume Conjecture, which is the driving force to enrich the theory of knots. Nowadays, there are various algorithms to compute Jones polynomials $J(q)$ itself. In particular, for computing values $J(e^{\frac{2\pi i}{n}})$ at $q = e^{\frac{2\pi i}{n}}$ ($n = 4, 5, 6, \cdots$), algorithms using quantum computations are known. Let's see one of the quantum algorithms for the case of the figure-eight knot.

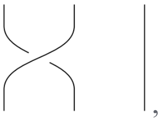
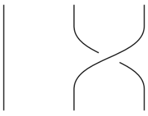
The figure-eight knot is one of the major knot in the study of knot theory. From the viewpoint of braids, the figure-eight knot is given by three strings and close those ends as follows (which end up with one loop):



Now, denote the crossings by symbols σ_1, σ_2 as follows:

$\sigma_1 :=$ , $\sigma_2 :=$ ,

Denote the crossings changing the over and the under as $\sigma_1^{-1}, \sigma_2^{-1}$:

$\sigma_1^{-1} :=$ , $\sigma_2^{-1} :=$ 

Then by observing the figure-eight knot from top to the bottom, we can represent the figure-eight knot by the symbol $\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1$.

Next, for σ_1, σ_2, correspond the unitary matrix as follows:

$$\begin{aligned} \sigma_1 &\mapsto A_1 := e^{\frac{\pi i}{8}} \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}, \\ \sigma_2 &\mapsto A_2 := \frac{1}{\sqrt{2}} \begin{pmatrix} e^{-\frac{\pi i}{8}} & e^{\frac{3\pi i}{8}} \\ e^{\frac{3\pi i}{8}} & e^{-\frac{\pi i}{8}} \end{pmatrix}. \end{aligned}$$

For $\sigma_1^{-1}, \sigma_2^{-1}$, we correspond those inverse matrices, namely adjoint matrices $A_1^\dagger, A_2^\dagger$. Then the figure-eight knot $\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1$ corresponds to the product of unitary matrices $A_2^\dagger A_1 A_2^\dagger A_1$. From knot theory, the trace (i.e. the sum of the $(1,1)$-component and the $(2,2)$-component) of $A_2^\dagger A_1 A_2^\dagger A_1$ coincides with -1. This value [*1] is $J(i)$, the Jones polynomial of the figure-eight knot at i.

Next, for simplicity, denote $\eta := 2\cos\frac{\pi}{5} = \frac{1+\sqrt{5}}{2}$. Then for σ_1, σ_2, we consider the following correspondence:

$$\begin{aligned} \sigma_1 &\mapsto B_1 := \begin{pmatrix} e^{-\frac{4\pi i}{5}} & 0 & 0 \\ 0 & e^{\frac{3\pi i}{5}} & 0 \\ 0 & 0 & e^{\frac{3\pi i}{5}} \end{pmatrix}, \\ \sigma_2 &\mapsto B_2 := \begin{pmatrix} \eta^{-1} e^{\frac{4\pi i}{5}} & \eta^{-\frac{1}{2}} e^{-\frac{3\pi i}{5}} & 0 \\ \eta^{-\frac{1}{2}} e^{-\frac{3\pi i}{5}} & -\eta^{-1} & 0 \\ 0 & 0 & e^{\frac{3\pi i}{5}} \end{pmatrix}. \end{aligned}$$

Put b_{ij} the (i,j)-component of the matrix product $B_2^\dagger B_1 B_2^\dagger B_1$, which corresponds with the figure-eight knot. Then, from knot theory, the value (which is called the *Markov trace* of $B_2^\dagger B_1 B_2^\dagger B_1$)

$$\eta^2 \cdot \frac{1}{\sin\frac{2\pi}{5} + \sin\frac{2\pi}{5} + \sin\frac{4\pi}{5}} \cdot \left(\sin\frac{2\pi}{5} \cdot b_{11} + \sin\frac{2\pi}{5} \cdot b_{22} + \sin\frac{4\pi}{5} \cdot b_{33} \right)$$

coincides with $-\sqrt{5}+1$. This value [*1] is $J(e^{\frac{2\pi i}{5}})$, the Jones polynomial of the figure-eight knot at $e^{\frac{2\pi i}{5}}$.

In a summary, when we understand the diagonal components of the product of unitary matrices, we can compute $J(i)$ and $J(e^{\frac{2\pi i}{5}})$. Of course, we can compute those products directly, but in this post, we will use the quantum algorithm, called the Hadamard test.

[*1]

The Jones polynomial $J(q)$ of the figure-eight knot is given by $q^2 - q + 1 - q^{-1} + q^{-2}$. The value of putting q as i and $e^{\frac{2\pi i}{5}}$ are respectively given as follows:

$$\begin{aligned} J(i) &= (-1) - i + 1 - (-i) + (-1) = -1 \\ J\left(e^{\frac{2\pi i}{5}}\right) &= \left(e^{\frac{4\pi i}{5}} + e^{-\frac{4\pi i}{5}}\right) + \left(e^{\frac{2\pi i}{5}} + e^{-\frac{2\pi i}{5}}\right) + 1 \\ &= 2\cos\frac{4\pi}{5} - 2\cos\frac{2\pi}{5} + 1 \\ &= 2 \cdot \left(-\frac{\sqrt{5}+1}{4}\right) - 2 \cdot \frac{\sqrt{5}-1}{4} + 1 \\ &= -\sqrt{5} + 1. \end{aligned}$$

Preliminaries of quantum computation

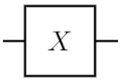
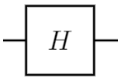
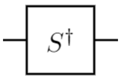
First, let me list the basis of quantum computation, which we will use in this post (also to check my understanding is correct).

In this post, the canonical vector is denoted by using the bra-ket notation as follows:

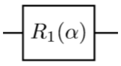
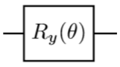
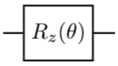
bra-ket notation	vector
$\|0\rangle$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$
$\|1\rangle$	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$
$\langle 0\|$	$\begin{pmatrix} 1 & 0 \end{pmatrix}$
$\langle 1\|$	$\begin{pmatrix} 0 & 1 \end{pmatrix}$

Let $|\psi\rangle := c_0|0\rangle + c_1|1\rangle = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}$, where c_0 and c_1 are complex numbers. Then $|\psi\rangle$ is called a **qubit** if c_0 and c_1 satisfies $|c_0|^2 + |c_1|^2 = 1$, where $|\cdot|$ denotes the absolute value of the complex number.

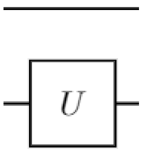
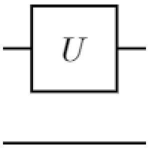
Following are the major quantum gates, which acts only to one qubit:

quantum gate	unitary matrix	bra-ket notation
	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\|1\rangle\langle 0 \| + \| 0\rangle\langle 1\|$
	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$	$\frac{\|0\rangle+\|1\rangle}{\sqrt{2}}\langle 0 \| + \frac{\|0\rangle-\|1\rangle}{\sqrt{2}}\langle 1\|$
	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}$	$\|0\rangle\langle 0 \| -i \| 1\rangle\langle 1\|$

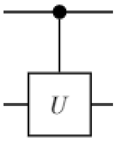
Following are the basic quantum gates representing major rotations on the Bloch sphere:

quantum gate	unitary matrix
	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$
	$\begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
	$\begin{pmatrix} e^{-\frac{i\theta}{2}} & 0 \\ 0 & e^{\frac{i\theta}{2}} \end{pmatrix}$

Next, let $U := \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ be a unitary matrix. Denote $I := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ the identity matrix. Quantum gates for two qubits are represented in matrices by the tensor product $\otimes$ as follows:

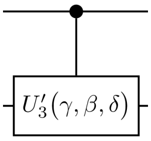
quantum gate	unitary matrix
	$I \otimes U = \begin{pmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{pmatrix}$
	$U \otimes I = \begin{pmatrix} a & 0 & b & 0 \\ 0 & a & 0 & b \\ c & 0 & d & 0 \\ 0 & c & 0 & d \end{pmatrix}$

For other major quantum gates for two qubits, we have **controlled gates**:

quantum gate	unitary matrix	bra-ket notation
 controlled-U	$\begin{pmatrix} I & O \\ O & U \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{pmatrix}$	$\|0\rangle\langle 0 \| \otimes I + \| 1\rangle\langle 1 \| \otimes U$

Actually, any controlled-U gates can be represented by $R_1(\alpha)$ and the following $cU_3(\gamma, \beta, \delta)$, where α, γ, β and δ are real numbers. In order to simplify explanations of this reason and latter discussions, we denote $U_3'(\gamma, \beta, \delta)$ the unitary matrix [*2]

$$\begin{pmatrix} e^{i(-\frac{\beta+\delta}{2})} \cos\frac{\gamma}{2} & -e^{i(-\frac{\beta-\delta}{2})} \sin\frac{\gamma}{2} \\ e^{i\cdot\frac{\beta-\delta}{2}} \sin\frac{\gamma}{2} & e^{i\cdot\frac{\beta+\delta}{2}} \cos\frac{\gamma}{2} \end{pmatrix}.$$

quantum gate	unitary matrix
 $cU_3(\gamma,\beta,\delta)$	$\begin{pmatrix} I & O \\ O & U_3'(\gamma,\beta,\delta) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i(-\frac{\beta+\delta}{2})}\cos\frac{\gamma}{2} & -e^{i(-\frac{\beta-\delta}{2})}\sin\frac{\gamma}{2} \\ 0 & 0 & e^{i\cdot\frac{\beta-\delta}{2}}\sin\frac{\gamma}{2} & e^{i\cdot\frac{\beta+\delta}{2}}\cos\frac{\gamma}{2} \end{pmatrix}$

The reason is as follows: for any unitary matrix U, we can write [*3]

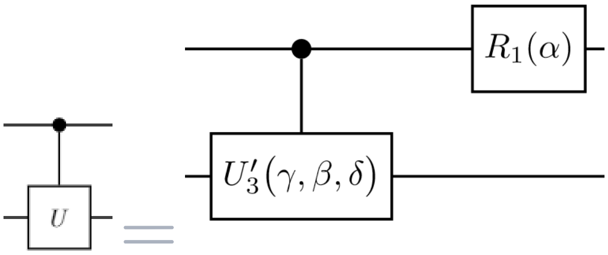
$$U = e^{i\alpha}\cdot U_3'(\gamma,\beta,\delta) = \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\alpha} \end{pmatrix}\begin{pmatrix} e^{i(-\frac{\beta+\delta}{2})}\cos\frac{\gamma}{2} & -e^{i(-\frac{\beta-\delta}{2})}\sin\frac{\gamma}{2} \\ e^{i\cdot\frac{\beta-\delta}{2}}\sin\frac{\gamma}{2} & e^{i\cdot\frac{\beta+\delta}{2}}\cos\frac{\gamma}{2} \end{pmatrix}.$$

So, since

$$\text{controlled-}\begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\alpha} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\alpha} & 0 \\ 0 & 0 & 0 & e^{i\alpha} \end{pmatrix} = \begin{pmatrix} 1\cdot I & 0 \\ 0 & e^{i\alpha}\cdot I \end{pmatrix} = R_1(\alpha)\otimes I$$

we obtain the following equation (be careful of the order of setting gates)：

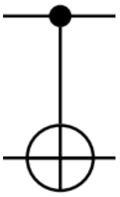
$$\text{controlled-}U = (R_1(\alpha)\otimes I)\cdot cU_3(\gamma,\beta,\delta).$$



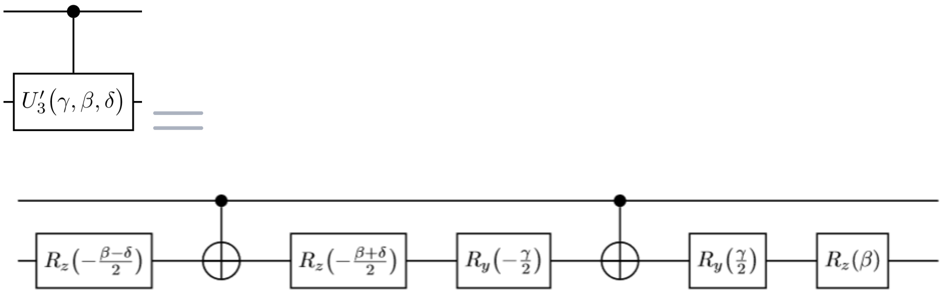
Moreover, $U_3'(\gamma,\beta,\delta)$ can be decomposed into X, $R_y(\theta)$ and $R_z(\theta)$ as follows:

$$U_3'(\gamma,\beta,\delta) = R_z(\beta)R_y\left(\frac{\gamma}{2}\right)\cdot X\cdot R_y\left(-\frac{\gamma}{2}\right)R_z\left(-\frac{\beta+\delta}{2}\right)\cdot X\cdot R_z\left(-\frac{\beta-\delta}{2}\right).$$

Since $R_z(\beta)R_y\left(\frac{\gamma}{2}\right)\cdot R_y\left(-\frac{\gamma}{2}\right)R_z\left(-\frac{\beta+\delta}{2}\right)\cdot R_z\left(-\frac{\beta-\delta}{2}\right) = I$, by using **CNOT**, namely the controlled-X,

quantum gate	unitary matrix
 CNOT	$\begin{pmatrix} I & O \\ O & X \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

we can decompose $cU_3(\gamma,\beta,\delta) = \text{controlled-}U_3'(\gamma,\beta,\delta)$ as follows (cf. [Nielsen-Chuang; Section 4.2]):



$U_3'(\gamma, \beta, \delta)$
$R_z(-\frac{\beta-\delta}{2})$
$R_z(-\frac{\beta+\delta}{2})$
$R_y(-\frac{\gamma}{2})$
$R_y(\frac{\gamma}{2})$
$R_z(\beta)$

We will use these equation in the latter discussion.

Finally, let us roughly see how the probability of measuring only one qubit is computed. For a qubit $|\psi\rangle = c_0|0\rangle + c_1|1\rangle = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}$, the **probablilty of measuring $|0\rangle$** is defined by $|c_0|^2$. This probability can be computed by using the norm of the vector $||\cdot||$ and the matrix $P_0 := |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ by $||P_0 \cdot |\psi\rangle||^2$. Following this computation, for the two qubits $|\psi_1\rangle \otimes |\psi_2\rangle$, the **probablilty of measuring $|0\rangle$ for the first (left) qubit** is computed by $||(P_0 \otimes I) \cdot (|\psi_1\rangle \otimes |\psi_2\rangle)||^2$. In this post, we denote this measurement by .

$*2$

The reason why I put the prime symbol $'$ on $U_3(\gamma, \beta, \delta)$ is that $U_3'(\gamma, \beta, \delta)$ is generally different from $U_3(\gamma, \beta, \delta) := \begin{pmatrix} \cos\frac{\gamma}{2} & -e^{i\delta}\sin\frac{\gamma}{2} \\ e^{i\beta}\sin\frac{\gamma}{2} & e^{i\cdot(\beta+\delta)}\cos\frac{\gamma}{2} \end{pmatrix}$, which is implemented in Qiskit. We have $U_3'(\gamma, \beta, \delta) = e^{i(-\frac{\beta+\delta}{2})}U_3(\gamma, \beta, \delta)$. Hence, $cU_3(\gamma, \beta, \delta) = \text{controlled-}U_3'(\gamma, \beta, \delta)$ and $\text{controlled-}U_3(\gamma, \beta, \delta)$ also differs in general.

$*3$

Let $U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Since, $|\det(U)| = 1, UU^\dagger = I$, we have

$$\begin{aligned} ad - bc &= e^{i\alpha'} && (1) \\ a\overline{c} + b\overline{d} &= 0 && (2) \\ |c| + |d| &= 1, && (3) \end{aligned}$$

where α' is a real number, and $^-$ represents the complex conjugate. By $(1) \times \overline{d} + (2) \times c$, we have

$$a(|c| + |d|) = \overline{d}e^{i\alpha'},$$

and by $(1) \times \overline{c} + (2) \times d$, we have

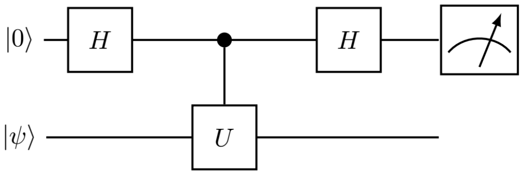
$$-b(|c| + |d|) = \overline{c}e^{i\alpha'},$$

and so by (3), we obtain $a = \overline{d}e^{i\alpha'}$, $b = -\overline{c}e^{i\alpha'}$. Again, by (3), c and d can be represented by $c = e^{i\theta_1}\sin\frac{\gamma}{2}$, $d = e^{i\theta_2}\cos\frac{\gamma}{2}$, where $\theta_1, \theta_2, \gamma$ are real numbers. Hence, by putting $\alpha' = 2\alpha, \theta_1 = \alpha + \frac{\beta-\delta}{2}, \theta_2 = \alpha + \frac{\beta+\delta}{2}$ we have

$$U = \begin{pmatrix} e^{i(\alpha'-\theta_2)}\cos\frac{\gamma}{2} & -e^{i(\alpha'-\theta_1)}\sin\frac{\gamma}{2} \\ e^{i\theta_1}\sin\frac{\gamma}{2} & e^{i\theta_2}\cos\frac{\gamma}{2} \end{pmatrix} = e^{i\alpha}\begin{pmatrix} e^{i(-\frac{\beta+\delta}{2})}\cos\frac{\gamma}{2} & -e^{i(-\frac{\beta-\delta}{2})}\sin\frac{\gamma}{2} \\ e^{i\cdot\frac{\beta-\delta}{2}}\sin\frac{\gamma}{2} & e^{i\cdot\frac{\beta+\delta}{2}}\cos\frac{\gamma}{2} \end{pmatrix} = \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\alpha} \end{pmatrix} U_3'(\gamma,\beta,\delta).$$

Hadamard test

Next, let's see what the Hadamard test is. There are two types of Hadamard test: computing real parts of diagonal components of a unitary matrix U, and computing imaginary parts of those. When we put $|\psi\rangle$ as either $|0\rangle$ or $|1\rangle$, the quantum circuit of the Hadamard test for the real part is given as follows:



Since $\langle 0|0\rangle = 1$, $\langle 0|1\rangle = 0$, by computing the state just before the measurement, we have

$$\begin{aligned} |0\rangle \otimes |\psi\rangle &\overset{H\otimes I}{\longmapsto} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |\psi\rangle \\ &\overset{\text{controlled-}U}{\longmapsto} \frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle + |1\rangle \otimes U|\psi\rangle) \\ &\overset{H\otimes I}{\longmapsto} \frac{1}{2}((|0\rangle + |1\rangle) \otimes |\psi\rangle + (|0\rangle - |1\rangle) \otimes U|\psi\rangle) \\ &= \frac{1}{2}(|0\rangle \otimes (I+U)|\psi\rangle + |1\rangle \otimes (I-U)|\psi))\rangle) \end{aligned}$$

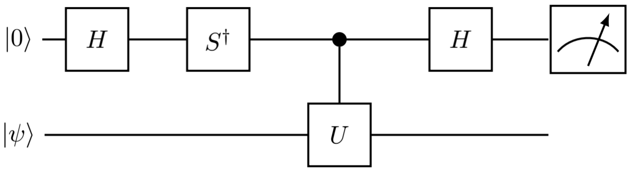
Now, let $p_{0,|\psi\rangle,\mathrm{Re}}$ be the probability of measuring $|0\rangle$ at the first (top) qubit. Then since U is unitary, we have the equation (diagonal component of $(U+U^\dagger)$) $= 2\cdot$ (real part of diagonal component of U), and so

$$\begin{aligned} p_{0,|\psi\rangle,\mathrm{Re}} &= \left\| (P_0 \otimes I)\cdot\left(\frac{1}{2}(|0\rangle \otimes (I+U)|\psi\rangle + |1\rangle \otimes (I-U)|\psi\rangle)\right)\right\|^2 \\ &= \left\|\frac{1}{2}(|0\rangle \otimes (I+U)|\psi\rangle)\right\|^2 \\ &= \left\|\frac{1}{2}\begin{pmatrix} 1\cdot(I+U)|\psi\rangle \\ 0\cdot(I+U)|\psi\rangle \end{pmatrix}\right\|^2 \\ &= \left\|\frac{1}{2}(I+U)|\psi\rangle\right\|^2 \\ &= \frac{1}{4}\langle\psi|(I+U)^\dagger(I+U)|\psi\rangle \\ &= \frac{1}{4}\langle\psi|(I+U^\dagger)(I+U)|\psi\rangle \\ &= \frac{1}{4}\langle\psi|(2I+(U+U^\dagger))|\psi\rangle \\ &= \frac{1}{2}\big(1+\mathrm{Re}(\langle\psi|U|\psi\rangle)\big). \end{aligned}$$

Hence, we have $\mathrm{Re}(\langle\psi|U|\psi\rangle) = 2p_{0,|\psi\rangle,\mathrm{Re}} - 1$, and so

$$\begin{aligned} &\text{Real part of } (1,1)\text{-component of } U = 2p_{0,|0\rangle,\mathrm{Re}} - 1, \\ &\text{Real part of } (2,2)\text{-component of } U = 2p_{0,|1\rangle,\mathrm{Re}} - 1. \end{aligned}$$

For the imaginary parts, the quantum circuit of the Hadamard test is given by as follows:



Similarly, let $p_{0,|\psi\rangle,\mathrm{Im}}$ be the probability of measuring $|0\rangle$ at the first (top) qubit. Then by the similar calculation, we have

$$\begin{aligned}\text{Imaginary part of } (1,1)\text{-component of } U &= 2p_{0,|0\rangle,\mathrm{Im}} - 1,\\ \text{Imaginary part of } (2,2)\text{-component of } U &= 2p_{0,|1\rangle,\mathrm{Im}} - 1.\end{aligned}$$

The Hadamard test plays an essential role when we can decompose a given unitary matrix into some meaningful unitary matrices. One of these cases is our goal in this post.

Compute approximate values at the imaginary unit

We are now ready to compute the approximate value of the Jones polynomial of the figure-eight knot at the imaginary unit i using Q#.

Our goal was: for unitary matrices

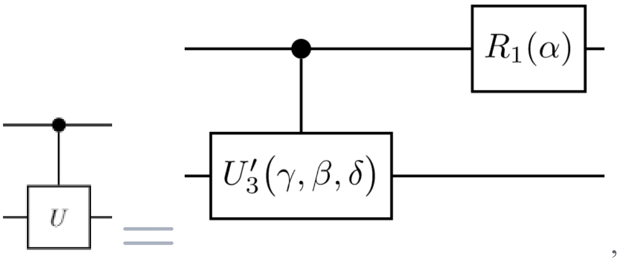
$$A_1 := e^{\frac{\pi i}{8}} \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}, \ A_2 := \frac{1}{\sqrt{2}} \begin{pmatrix} e^{-\frac{\pi i}{8}} & e^{\frac{3\pi i}{8}} \\ e^{\frac{3\pi i}{8}} & e^{-\frac{\pi i}{8}} \end{pmatrix},$$

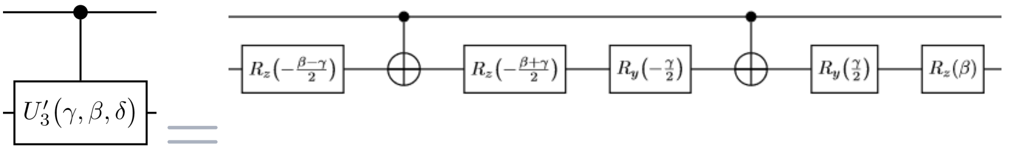
we would like to check that the value of the trace of $A_2^\dagger A_1 A_2^\dagger A_1$ approximately coincides with -1.

In Q#, quantum gates $R_1(\alpha)$, $R_y(\theta)$, $R_z(\theta)$ and CNOT are already implemented. Recall that any unitary matrix U can be parametrized by four real values α, γ, β and δ as follows:

$$U = e^{i\alpha} \cdot U_3'(\gamma, \beta, \delta) = \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\alpha} \end{pmatrix} \begin{pmatrix} e^{i(-\frac{\beta+\delta}{2})} \cos\frac{\gamma}{2} & -e^{i(-\frac{\beta-\delta}{2})} \sin\frac{\gamma}{2} \\ e^{i\cdot\frac{\beta-\delta}{2}} \sin\frac{\gamma}{2} & e^{i\cdot\frac{\beta+\delta}{2}} \cos\frac{\gamma}{2} \end{pmatrix}.$$

Since we have the following two equations of controlled gates,




$U_3'(\gamma, \beta, \delta)$
$=$
$R_z(-\frac{\beta-\gamma}{2})$
$R_z(-\frac{\beta+\gamma}{2})$
$R_y(-\frac{\gamma}{2})$
$R_y(\frac{\gamma}{2})$
$R_z(\beta)$
,

we only have to find four real parameters α, γ, β and δ for each matrices to implement controlled gates for any unitary matrices.

First, let's decompose the unitary matrices using $R_1(\alpha)$ and $cU_3(\gamma,\beta,\delta)$. Note that A_1 and A_2 can be represented as follows:

$$A_1 = e^{\frac{\pi i}{8}}\begin{pmatrix} e^{0\cdot\pi i} & 0 \\ 0 & e^{-\frac{\pi i}{2}}\end{pmatrix} = e^{-\frac{\pi i}{8}}\begin{pmatrix} e^{\frac{\pi i}{4}} & 0 \\ 0 & e^{-\frac{\pi i}{4}}\end{pmatrix} = e^{-\frac{\pi i}{8}}\begin{pmatrix} e^{i\cdot(-\frac{1}{2}(-\frac{\pi}{2}+0))}\cos 0 & -e^{i\cdot(-\frac{1}{2}(-\frac{\pi}{2}-0))}\sin 0 \\ e^{i\cdot\frac{1}{2}(-\frac{\pi}{2}-0)}\sin 0 & e^{i\cdot\frac{1}{2}(-\frac{\pi}{2}+0)}\cos 0\end{pmatrix},$$

$$A_2 = e^{-\frac{\pi i}{8}}\begin{pmatrix} e^{0\cdot\pi i}\cdot\frac{1}{\sqrt{2}} & e^{\frac{\pi i}{2}}\cdot\frac{1}{\sqrt{2}} \\ e^{\frac{\pi i}{2}}\cdot\frac{1}{\sqrt{2}} & e^{0\cdot\pi i}\cdot\frac{1}{\sqrt{2}}\end{pmatrix} = e^{-\frac{\pi i}{8}}\begin{pmatrix} e^{i\cdot(-\frac{1}{2}(\frac{\pi}{2}-\frac{\pi}{2}))}\cos(\frac{1}{2}\cdot\frac{\pi}{2}) & -e^{i\cdot(-\frac{1}{2}(\frac{\pi}{2}+\frac{\pi}{2}))}\sin(\frac{1}{2}\cdot\frac{\pi}{2}) \\ e^{i\cdot\frac{1}{2}(\frac{\pi}{2}+\frac{\pi}{2})}\sin(\frac{1}{2}\cdot\frac{\pi}{2}) & e^{i\cdot\frac{1}{2}(\frac{\pi}{2}-\frac{\pi}{2})}\cos(\frac{1}{2}\cdot\frac{\pi}{2})\end{pmatrix}.$$

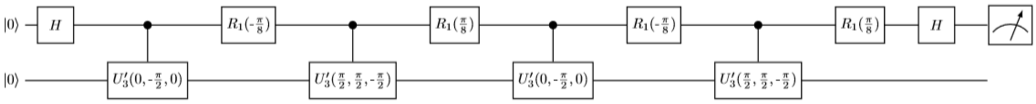
Since $\overline{e^{i\theta}} = e^{-i\theta}$, where $^{-}$ means the complex conjugate, the adjoint matrix $A_2^\dagger = {}^{t}\overline{A_2}$ (the complex conjugate and the transpose) of A_2 is represented as follows:

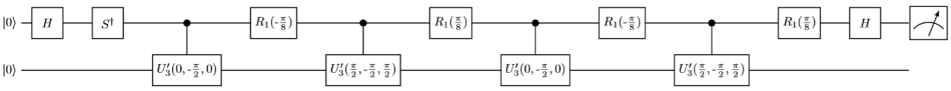
$$A_2^\dagger = e^{\frac{\pi i}{8}}\begin{pmatrix} e^{0\cdot\pi i}\cdot\frac{1}{\sqrt{2}} & e^{-\frac{\pi i}{2}}\cdot\frac{1}{\sqrt{2}} \\ e^{-\frac{\pi i}{2}}\cdot\frac{1}{\sqrt{2}} & e^{0\cdot\pi i}\cdot\frac{1}{\sqrt{2}}\end{pmatrix} = e^{\frac{\pi i}{8}}\begin{pmatrix} e^{i\cdot(-\frac{1}{2}(-\frac{\pi}{2}+\frac{\pi}{2}))}\cos(\frac{1}{2}\cdot\frac{\pi}{2}) & -e^{i\cdot(-\frac{1}{2}(-\frac{\pi}{2}-\frac{\pi}{2}))}\sin(\frac{1}{2}\cdot\frac{\pi}{2}) \\ e^{i\cdot\frac{1}{2}(-\frac{\pi}{2}-\frac{\pi}{2})}\sin(\frac{1}{2}\cdot\frac{\pi}{2}) & e^{i\cdot\frac{1}{2}(-\frac{\pi}{2}+\frac{\pi}{2})}\cos(\frac{1}{2}\cdot\frac{\pi}{2})\end{pmatrix}.$$

So, by using $R_1(\alpha)$ and $cU_3(\gamma,\beta,\delta)$, the controlled gates of A_1 and $A_2^\dagger$ can be decomposed as follows:

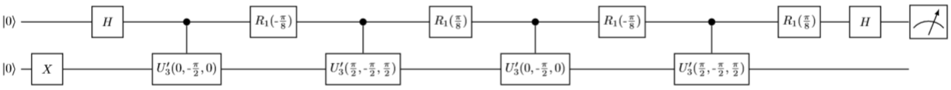
$$\begin{aligned}\text{controlled-}A_1 &= \left(R_1\left(-\frac{\pi}{8}\right)\otimes I_2\right)\cdot cU_3\left(0,-\frac{\pi}{2},0\right),\\ \text{controlled-}A_2^\dagger &= \left(R_1\left(\frac{\pi}{8}\right)\otimes I_2\right)\cdot cU_3\left(\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2}\right).\end{aligned}$$

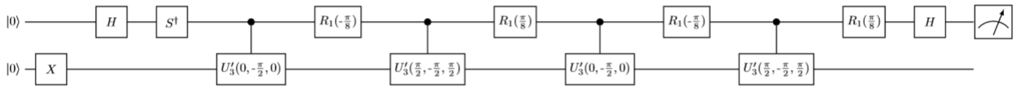
Therefore, Hadamard tests of computing the real part and the imaginary part of the $(1,1)$-component of $A_2^\dagger A_1 A_2^\dagger A_1$ are respectively given as follows:


|0⟩
|0⟩
H
$R_1(-\frac{\pi}{8})$
$R_1(\frac{\pi}{8})$
$R_1(-\frac{\pi}{8})$
$R_1(\frac{\pi}{8})$
H
$U_3'(0,-\frac{\pi}{2},0)$
$U_3'(\frac{\pi}{2},\frac{\pi}{2},-\frac{\pi}{2})$
$U_3'(0,-\frac{\pi}{2},0)$
$U_3'(\frac{\pi}{2},\frac{\pi}{2},-\frac{\pi}{2})$

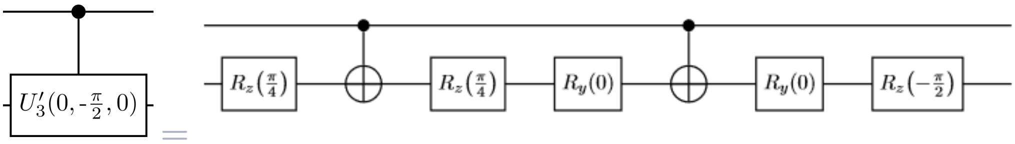

|0⟩
|0⟩
H
$S^\dagger$
$R_1(-\frac{\pi}{8})$
$R_1(\frac{\pi}{8})$
$R_1(-\frac{\pi}{8})$
$R_1(\frac{\pi}{8})$
H
$U_3'(0,-\frac{\pi}{2},0)$
$U_3'(\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2})$
$U_3'(0,-\frac{\pi}{2},0)$
$U_3'(\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2})$

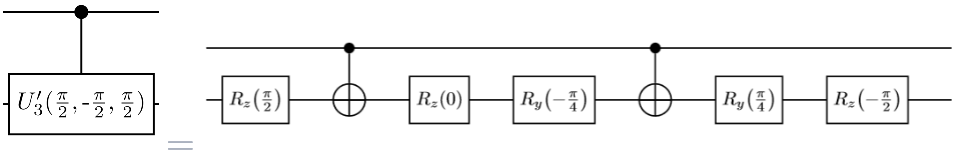
On the other hand, since $X|0\rangle = |1\rangle$, Hadamard tests of computing the real part and the imaginary part of the $(2,2)$-component of $A_2^\dagger A_1 A_2^\dagger A_1$ are respectively given as follows:


|0⟩
|0⟩
X
H
$R_1(-\frac{\pi}{8})$
$R_1(\frac{\pi}{8})$
$R_1(-\frac{\pi}{8})$
$R_1(\frac{\pi}{8})$
H
$U_3'(0,-\frac{\pi}{2},0)$
$U_3'(\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2})$
$U_3'(0,-\frac{\pi}{2},0)$
$U_3'(\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2})$


|0⟩
|0⟩
X
H
$S^\dagger$
$R_1(-\frac{\pi}{8})$
$R_1(\frac{\pi}{8})$
$R_1(-\frac{\pi}{8})$
$R_1(\frac{\pi}{8})$
H
$U_3'(0,-\frac{\pi}{2},0)$
$U_3'(\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2})$
$U_3'(0,-\frac{\pi}{2},0)$
$U_3'(\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2})$

Moreover, using $R_y(\theta)$, $R_z(\theta)$ and CNOT, we can decompose $cU_3\left(0,-\frac{\pi}{2},0\right)$ and $cU_3\left(\frac{\pi}{2},-\frac{\pi}{2},\frac{\pi}{2}\right)$ as follows:





We are now ready to write codes in Q# and C#. In the order as above, we can honestly implement Hadamard tests in Q# as follows:

```
namespace Quantum.Jones_figEight_4th
{
    open Microsoft.Quantum.Primitive;
    open Microsoft.Quantum.Canon;
    open Microsoft.Quantum.Extensions.Math; // PI()

    operation Set (desired: Result, q: Qubit) : Unit
    {
        let current = M(q);
        if (desired != current)
        {
            X(q);
        }
    }

    operation HadamardTest_figEight_4th (count : Int, initial :
Result, index : Int) : (Int, Int)
    {
        mutable numZeros = 0;

        using (qubits = Qubit[2])
        {
            for (test in 1..count)
            {
                // initial state: |00>
                Set (Zero, qubits[0]);
                Set (Zero, qubits[1]);

                if (index / 2 == 1)
                {
                    X(qubits[1]);
                }

                H(qubits[0]);

                if (index % 2 == 1)
                {
                    Adjoint(S)(qubits[0]);
                }

                // controlled-A1
                Rz(0.25 * PI(), qubits[1]);
                CNOT(qubits[0], qubits[1]);
                Rz(0.25 * PI(), qubits[1]);
                Ry(0.0, qubits[1]);
                CNOT(qubits[0], qubits[1]);
                Ry(0.0, qubits[1]);
                Rz(-0.50 * PI(), qubits[1]);
                R1(-0.125 * PI(), qubits[0]);

                // controlled-A2†
                Rz(0.50 * PI(), qubits[1]);
                CNOT(qubits[0], qubits[1]);
                Rz(0.0, qubits[1]);
                Ry(-0.25 * PI(), qubits[1]);
                CNOT(qubits[0], qubits[1]);
                Ry(0.25 * PI(), qubits[1]);
                Rz(-0.50 * PI(), qubits[1]);
                R1(0.125 * PI(), qubits[0]);

                // controlled-A1
                Rz(0.25 * PI(), qubits[1]);
                CNOT(qubits[0], qubits[1]);
                Rz(0.25 * PI(), qubits[1]);
                Ry(0.0, qubits[1]);
                CNOT(qubits[0], qubits[1]);
                Ry(0.0, qubits[1]);
                Rz(-0.50 * PI(), qubits[1]);
                R1(-0.125 * PI(), qubits[0]);

                // controlled-A2†
                Rz(0.50 * PI(), qubits[1]);
                CNOT(qubits[0], qubits[1]);
                Rz(0.0, qubits[1]);
                Ry(-0.25 * PI(), qubits[1]);
                CNOT(qubits[0], qubits[1]);
                Ry(0.25 * PI(), qubits[1]);
                Rz(-0.50 * PI(), qubits[1]);
                R1(0.125 * PI(), qubits[0]);

                H(qubits[0]);

                let res = M(qubits[0]);

                // count |0>
                if (res == Zero)
                {
                    set numZeros = numZeros + 1;
                }
            }

            ResetAll(qubits);
        }

        // number of |0> and |1>
        return (numZeros, count - numZeros);
    }
}
```

In C#, we perform classical computations. Namely, we compute the probabilities p_0s of measurng $|0\rangle$ when the number of shos are 1000, and create the list of $2p_0 - 1$ representing the real and imaginary parts of diagonal components of matrices we want to find, and lastly, compute the real and imaginary parts of the trace we want to find. To wrap up, we can implement the C# code associated with the above Q# code as follows:

```
using System;
using System.Collections.Generic; // List
using Microsoft.Quantum.Simulation.Core;
using Microsoft.Quantum.Simulation.Simulators;

namespace Quantum.Jones_figEight_4th
{
    class Driver
    {
        static void Main(string[] args)
        {
            using (var qsim = new QuantumSimulator())
            {
                var listVals = new List<double>();

                for (int i = 0; i < 4; i++)
                {
                    var (num0s, num1s) =
HadamardTest_figEight_4th.Run(qsim, 1000, Result.Zero, i).Result;
                    Console.WriteLine($"{i}: 0s={num0s,-4} 1s=
{num1s,-4}");

                    double p0 = num0s / 1000.0;
                    listVals.Add(2 * p0 - 1);
                }

                Console.WriteLine($"Re: {listVals[0] +
listVals[2],-4}");
                Console.WriteLine($"Im: {listVals[1] +
listVals[3],-4}");
            }

            Console.ReadKey();
        }
    }
}
```

By (building and) running these codes together, we obtain the following result:

```
0: 0s=264  1s=736
1: 0s=748  1s=252
2: 0s=222  1s=778
3: 0s=242  1s=758
Re: -1.028
Im: -0.02
```

Therefore, we can see that the nearest integer ^{*4} of the trace of $A_2^\dagger A_1 A_2^\dagger A_1$ is -1, which coincides with $J(i)$, the value of the Jones polynomial of the figure-eight knot at i.

$*4$

For a general knot K, since the value of the Jones polynomial $J_K(i)$ of K at i coincides with $(-1)^{\mathrm{Arf}(K)}$, where $\mathrm{Arf}(K)$ (called the *Arf invaiant* of K) takes value with either 0 or 1 (cf. [Murakami]), the value $J_K(i)$ is an integer. Hence, we only need to find the nearest integer of the result we obtained. By the result, we can compute that $\mathrm{Arf}(K) = 1$, when K is a figure-eight knot.

Compute approximate values at the 5th root of unity

Lastly, we'll try to compute the approximate value of $J(e^{\frac{2\pi i}{5}})$, the Jones polynomial of the figure-eight knot the 5th root of unity, using the similar method as before. Our goal was to compute the value

$$\eta^2 \cdot \frac{1}{\sin\frac{2\pi}{5} + \sin\frac{2\pi}{5} + \sin\frac{4\pi}{5}} \cdot \left(\sin\frac{2\pi}{5} \cdot b_{11} + \sin\frac{2\pi}{5} \cdot b_{22} + \sin\frac{4\pi}{5} \cdot b_{33} \right),$$

where $\eta := 2\cos\frac{\pi}{5} = \frac{1+\sqrt{5}}{2}$, and b_{ii} is the i-th diagonal component of the product of unitary matrices $B_2^\dagger B_1 B_2^\dagger B_1$, where

$$B_1 := \begin{pmatrix} e^{-\frac{4\pi i}{5}} & 0 & 0 \\ 0 & e^{\frac{3\pi i}{5}} & 0 \\ 0 & 0 & e^{\frac{3\pi i}{5}} \end{pmatrix}, \ B_2 := \begin{pmatrix} \eta^{-1} e^{\frac{4\pi i}{5}} & \eta^{-\frac{1}{2}} e^{-\frac{3\pi i}{5}} & 0 \\ \eta^{-\frac{1}{2}} e^{-\frac{3\pi i}{5}} & -\eta^{-1} & 0 \\ 0 & 0 & e^{\frac{3\pi i}{5}} \end{pmatrix}.$$

First, for the $(3, 3)$-component b_{33} of $B_2^\dagger B_1 B_2^\dagger B_1$, we directly compute the product

$$\overline{e^{\frac{3\pi i}{5}}} \cdot e^{\frac{3\pi i}{5}} \cdot \overline{e^{\frac{3\pi i}{5}}} \cdot e^{\frac{3\pi i}{5}} = e^{-\frac{3\pi i}{5}} \cdot e^{\frac{3\pi i}{5}} \cdot e^{-\frac{3\pi i}{5}} \cdot e^{\frac{3\pi i}{5}} = 1.$$

In order to compute the rest of diagonal components, b_{11} and b_{22}, we consider 2×2 unitary matrices B_1' and B_1' obtained by removing the third row and column from B_1 and B_2:

$$B_1' := \begin{pmatrix} e^{-\frac{4\pi i}{5}} & 0 \\ 0 & e^{\frac{3\pi i}{5}} \end{pmatrix}, \ B_2' := \begin{pmatrix} \eta^{-1} e^{\frac{4\pi i}{5}} & \eta^{-\frac{1}{2}} e^{-\frac{3\pi i}{5}} \\ \eta^{-\frac{1}{2}} e^{-\frac{3\pi i}{5}} & -\eta^{-1} \end{pmatrix}.$$

Then, note that b_{11} and b_{22} coincide respectively with the $(1, 1)$-component and the $(2, 2)$-component of the product of matrices ${B_2'}^\dagger B_1' {B_2'}^\dagger B_1'$.

Here, we'll use more efficient way than before to implement controlled gates. Note that in Q#, implementations of the adjoint gate and the controlled gate of any quantum gate are prepared. So, it suffices to decompose only B_1' and B_2' (not ${B_2'}^\dagger$ nor those controlled gates!).

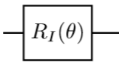
In order to implement general 2×2 unitary matrices, we use the following equation, which we saw in the preliminaries:

$$U_3'(\gamma, \beta, \delta) = R_z(\beta) R_y\left(\frac{\gamma}{2}\right) \cdot X \cdot R_y\left(-\frac{\gamma}{2}\right) R_z\left(-\frac{\beta+\delta}{2}\right) \cdot X \cdot R_z\left(-\frac{\beta-\delta}{2}\right).$$

Since $X \cdot R_y\left(-\frac{\gamma}{2}\right) R_z\left(-\frac{\beta+\delta}{2}\right) \cdot X = R_y\left(\frac{\gamma}{2}\right) R_z\left(\frac{\beta+\delta}{2}\right)$, we can decompose $U_3'(\gamma, \beta, \delta)$ into quantum gates $R_y(\theta)$ and $R_z(\theta)$ as follows:

$$U_3'(\gamma, \beta, \delta) = R_z(\beta) R_y(\gamma) R_z(\delta).$$

In general, 2×2 unitary matrices are represented as $e^{i\alpha} U_3'(\gamma, \beta, \delta)$ and so we have to consider how to represent the multiplication of $e^{i\alpha}$ into a quantum gate. In order to do this, we use the following gate, which is already implemented in Q# as `R(PauliI, theta, qubit)` (Note that we have to change the sign and divide θ by 2):

quantum gate	unitary matrix
	$\begin{pmatrix} e^{-\frac{i\theta}{2}} & 0 \\ 0 & e^{-\frac{i\theta}{2}} \end{pmatrix}$

Now we are ready to implement. Note that we can represent B_1' and B_1' as follows:

$$B_1' = e^{-\frac{\pi i}{10}} \begin{pmatrix} e^{-\frac{7\pi i}{10}} & 0 \\ 0 & e^{\frac{7\pi i}{10}} \end{pmatrix} = e^{-\frac{\pi i}{10}} \begin{pmatrix} e^{i \cdot (-\frac{1}{2}(\frac{7\pi}{5}+0))} \cos 0 & -e^{i \cdot (-\frac{1}{2}(\frac{7\pi}{5}-0))} \sin 0 \\ e^{i \cdot \frac{1}{2}(\frac{7\pi}{5}-0)} \sin 0 & e^{i \cdot \frac{1}{2}(\frac{7\pi}{5}+0)} \cos 0 \end{pmatrix},$$

$$B_2' = e^{\frac{9\pi i}{10}} \begin{pmatrix} e^{-\frac{\pi i}{10}} \cdot \eta^{-1} & e^{-\frac{3\pi i}{2}} \cdot \eta^{-\frac{1}{2}} \\ e^{-\frac{3\pi i}{2}} \cdot \eta^{-\frac{1}{2}} & e^{\frac{\pi i}{10}} \cdot \eta^{-1} \end{pmatrix} = e^{\frac{9\pi i}{10}} \begin{pmatrix} e^{i \cdot (-\frac{1}{2}(-\frac{7\pi}{5}+\frac{8\pi}{5}))} \cdot \eta^{-1} & e^{i \cdot (-\frac{1}{2}(-\frac{7\pi}{5}-\frac{8\pi}{5}))} \cdot \eta^{-\frac{1}{2}} \\ e^{i \cdot \frac{1}{2}(-\frac{7\pi}{5}-\frac{8\pi}{5}))} \cdot \eta^{-\frac{1}{2}} & e^{i \cdot \frac{1}{2}(-\frac{7\pi}{5}+\frac{8\pi}{5})} \cdot \eta^{-1} \end{pmatrix},$$

Here, we will use the approximate values
$\eta^{-1} = \frac{\sqrt{5}-1}{2} = 0.618\cdots \approx \cos 51.8^\circ = \cos\left(\frac{1}{2}\cdot\frac{103.6\pi}{180}\right)$ and
$\eta^{-\frac{1}{2}} \approx \sin 51.8^\circ = \sin\left(\frac{1}{2}\cdot\frac{103.6\pi}{180}\right)$. Then B_1' and B_2' are decomposed as follows:

$$
\begin{aligned}
B_1' &= R_I\left(\frac{\pi}{5}\right)\cdot U_3'\left(0, \frac{7}{5}\pi, 0\right), \\
B_2' &\approx R_I\left(-\frac{9\pi}{5}\right)\cdot U_3'\left(\frac{103.6\pi}{180}, -\frac{7\pi}{5}, \frac{8\pi}{5}\right).
\end{aligned}
$$

To wrap up, we can implement the code more legibly in Q# as follows:

```
namespace Quantum.Jones_figEight_5th
{
    open Microsoft.Quantum.Primitive;
    open Microsoft.Quantum.Canon;
    open Microsoft.Quantum.Extensions.Math;

    operation Set (desired: Result, q: Qubit) : Unit
    {
        let current = M(q);
        if (desired != current)
        {
            X(q);
        }
    }

    // U3'(γ, β, δ)
    operation U3p (gamma : Double, beta : Double, delta : Double, q
: Qubit) : Unit
    {
        body (...)
        {
              Rz(delta, q);
              Ry(gamma, q);
              Rz(beta, q);
        }

        adjoint auto;
        controlled auto;
        controlled adjoint auto;
    }

    // B1'
    operation B1p (q : Qubit) : Unit
    {
        body (...)
        {
              U3p(0.0, 1.40 * PI(), 0.0, q);
              R(PauliI, 0.20 * PI(), q);
        }

        adjoint auto;
        controlled auto;
        controlled adjoint auto;
    }

    // B2'
    operation B2p (q : Qubit) : Unit
    {
        body (...)
        {
              U3p((103.6 / 180.0) * PI(), -1.40 * PI(), 1.60 * PI(),
q);

              R(PauliI, -1.80 * PI(), q);
        }
        adjoint auto;
        controlled auto;
        controlled adjoint auto;
    }


    operation HadamardTest_figEight_5th (count : Int, initial :
Result, index : Int) : Int
    {
        mutable numZeros = 0;

        using (qubits = Qubit[2])
        {
            for (test in 1..count)
            {
                Set (Zero, qubits[0]);
                Set (Zero, qubits[1]);

                if (index / 2 == 1)
                {
                    X(qubits[1]);
                }

                H(qubits[0]);

                if (index % 2 == 1)
                {
                    Adjoint(S)(qubits[0]);
                }

                Controlled(B1p)([qubits[0]], qubits[1]);
                Controlled(Adjoint(B2p))([qubits[0]], qubits[1]);
                Controlled(B1p)([qubits[0]], qubits[1]);
                Controlled(Adjoint(B2p))([qubits[0]], qubits[1]);

                H(qubits[0]);

                let res = M(qubits[0]);

                if (res == Zero)
                {
                    set numZeros = numZeros + 1;
                }
            }

            ResetAll(qubits);
        }

        return numZeros;
    }
}
```

We would like to see that the first four digits of the result coincides with those of the value $J\left(e^{\frac{2\pi i}{5}}\right)$ of the Jones polynomial. In order to obtain more accurate values, we will iterate the above operation 1000 times, and take those averages. The implementation in C# is as follows:

```
using System;
using System.Collections.Generic; // List
using Microsoft.Quantum.Simulation.Core;
using Microsoft.Quantum.Simulation.Simulators;

namespace Quantum.Jones_figEight_5th
{
    class Driver
    {
        static void Main(string[] args)
        {
            using (var qsim = new QuantumSimulator())
            {
                var listVals = new List<double>();

                for (int i = 0; i < 4; i++)
                {
                    int sum0s = 0;

                    for (int j = 0; j < 1000; j++)
                    {
                        var num0s =
HadamardTest_figEight_5th.Run(qsim, 1000, Result.Zero, i).Result;
                        sum0s += (int)num0s;
                    }

                    double avgNum0s = sum0s / 1000.0;
                    Console.WriteLine($"{i}: avgNum0s=
{avgNum0s,-4}");

                    double p0 = avgNum0s / 1000.0;
                    listVals.Add(2 * p0 - 1);
                }

                double eta = 2.0 * Math.Cos(0.20 * Math.PI);
                double  s2 = Math.Sin(0.40 * Math.PI);
                double  s4 = Math.Sin(0.80 * Math.PI);

                Console.WriteLine($"Re: {eta * eta / (s2 + s2 + s4)
* (s2 * listVals[0] + s2 * listVals[2] + s4 * 1.0),-4}");
                Console.WriteLine($"Im: {eta * eta / (s2 + s2 + s4)
* (s2 * listVals[1] + s2 * listVals[3]),-4}");
            }

            Console.ReadKey();
        }
    }
}
```

Then, the result is as follows (on my cheap PC, this took about 4 hours):

```
0: avgNum0s=36.291
1: avgNum0s=612.42
2: avgNum0s=36.442
3: avgNum0s=388.01
Re: -1.2365000112501
Im: 0.000859999999999868
```

Therefore, we obtain the approximate value of $J\left(e^{\frac{2\pi i}{5}}\right)$, which equals to $(-\sqrt{5}+1)+0\cdot i=-1.23606\cdots$.

References

・D. Aharonov, V. Jones, and Z. Landau, A polynomial quantum algorithm for approximating the Jones polynomial, arXiv:quant-ph/0511096v2.

・B. Field and T. Simula, Introduction to topological quantum computation with non-Abelian anyons, arXiv:quant-ph/1802.06176v2.

・H. Murakami, A recursive calculation of the Arf invariant of a link, J. Math. Soc. Japan 38 (1986) 335–338.

・M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press (2010).

Acknowledgment

I would especially like to thank OpenQL community for inviting me to the field of quantum information.