Sevag Gharibian

Lecture 3: Bounded error quantum polynomial time (BQP)

"An algorithm must be seen to be believed, and the best way to learn what an algorithm is all about is to try it."
— Donald Knuth

Contents

Introduction. With our required complexity theory and quantum computation refreshers covered, we now focus on the cornerstones of quantum complexity theory — "quantum P" and "quantum NP". Defining the former of these will be arguably straightforward[1], but the same cannot be said for quantum "NP" — indeed, at present there is no "unique" definition of quantum NP, and for all we know, the various definitions currently in the literature may indeed be distinct entities.

This lecture begins by introducing the classical class BPP (Bounded-Error Probabilistic Polynomial Time), followed by BQP (i.e. "quantum Promise-BPP"). The lecture will introduce various fundamental concepts throughout, such as applications of norms to study error propagation in quantum circuits, as well as the notion of a quantum universal gate set. Note that while the quote atop this lecture is quite "hands-on" in nature, unfortunately we as a field are generally not in a position yet to simply "try out" quantum algorithms, at least on a large scale; indeed, we still have difficulty convincing ourselves the experiments we *do* run are doing what we expect them to do! (Such verification is a key issue in so-called "quantum supremacy" proposals, which will be discussed towards the end of the course.)

1 BPP

Recall from Lecture 2 that quantum computation is, in a rigorous sense, inherently probabilistic (i.e. we apply a unitary map to our input state, and then measure in the standard basis, yielding a probability distribution over outcomes). This raises the question of whether P is really the correct starting point for generalizing to BQP, since in stark contrast, P is about deterministic computation. A more suitable entry point seems to instead be *Bounded-Error Probabilistic Polynomial Time (BPP)*, which we now define.

Definition 1 (Bounded-Error Probabilistic Polynomial Time (BPP))**.** *A language $L \subseteq \{0,1\}^*$ is in BPP if there exists a (deterministic) TM M and fixed polynomials $s_L, r_L : \mathbb{N} \mapsto \mathbb{R}^+$, such that for any input $x \in \{0,1\}^n$, M takes in string $y \in \{0,1\}^{s_L(n)}$, halts in at most $O(r_L(n))$ steps, and:*

[1]This is a borderline false statement; we will need highly non-trivial facts such as the Solovay-Kitaev theorem to make "quantum P" well-defined. But at least at a high level, the definition of "quantum P" is what one might expect in hindsight.

- *(Completeness/YES case) If $x \in L$, then for at least $2/3$ of the choices of $y \in \{0,1\}^{s_L(n)}$, M accepts.*
- *(Soundness/NO case) If $x \notin L$, then for at most $1/3$ of the choices of $y \in \{0,1\}^{s_L(n)}$, M accepts.*

We say M accepts *(*rejects*) x in the YES case (NO case).*

There are three remarks worth making here. First, the definition of BPP looks suspiciously like NP — indeed, in NP we also had strings y, only we called them "proofs".

Exercise. How does the definition of BPP differ from that of NP?

The standard way to interpret y for BPP is as a uniformly random string over $\{0,1\}^n$. In other words, one thinks of a BPP machine M as a standard TM that also takes in a random string y as part of its input. With this view, in a YES case, M accepts with probability at least $2/3$, and in the NO case, M accepts with probability at most $1/3$. Second, under standard derandomization conjectures, it is generally believed that P = BPP. Proving this, however, is a longstanding open question.

Third, as phrased, BPP is a class of decision problems, i.e. every input x is either in L or not in L. This has a peculiar side-effect — on *any* input x, the machine M must accept or reject with probability at least $2/3$. It turns out that this subtle restriction is unnatural when moving to the quantum setting. Thus, BPP is *not* yet the correct starting point for generalizing to the quantum setting. Instead, we first need to move to *promise problems*.

1.1 Syntactic versus semantic classes, PromiseBPP, and what will really be PromiseBQP

To motivate PromiseBPP, we discuss syntactic versus semantic complexity classes.

Syntactic versus semantic classes. There is a subtle, but important difference, between how (say) P and NP are defined, as opposed to BPP. Namely, if we fix an appropriate encoding of TMs into strings, such that TM M has encoding $\langle M \rangle \in \{0,1\}^*$, then given a string $x \in \{0,1\}^*$, one can easily check if $x = \langle M \rangle$ for some P machine M. (For example, the encoding can explicitly state a threshold in terms of number of steps, after which the machine forcibly halts and outputs 0 or 1.) Complexity classes with this property are called *syntactic*. BPP, on the other hand, does not appear to have this property — how could any reasonable encoding of a TM M into strings encode the property that on all inputs, M accepts or rejects with probability at least $2/3$? (Indeed, deciding whether a TM has this property is undecidable, which follows from Rice's Theorem[2].) In other words, given $x \in \{0,1\}^*$, we cannot even decide whether x is a valid encoding of some BPP machine M!

Let us see how this causes problems. Consider the set of strings

$$L = \big\{ \langle M, x \rangle \mid M \text{ is a P machine which accepts } x \in \{0,1\}^* \big\}.$$

Clearly, $L \in$ P, since a P machine can check if $\langle M \rangle$ encodes a valid P machine first, and if so, run M on x and output M's answer. Now, consider instead

$$A = \big\{ \big\langle M, x, 1^t \big\rangle \mid M \text{ is a BPP machine which accepts } x \in \{0,1\}^* \text{ in at most } t \text{ steps} \big\}.$$

Since BPP is a semantic class (i.e. we don't know how to check if $\langle M \rangle$ is a valid encoding of a BPP machine), A is *not* obviously in BPP. (Indeed, A is actually #P-complete, and thus unlikely to be in BPP!)

[2]Rice's Theorem says that, for any "non-trivial" property P, deciding whether a given TM M has property P is undecidable. Here, "non-trivial" means there exists at least one TM with property P, and not all TMs have property P.

Exercise. Why doesn't the naive strategy we used for L show $A \in \text{BPP}$? In other words, why can a BPP machine not just try to simulate M on x?

This is a rather ridiculous state of affairs. Certainly the problem encoded by A is rather natural, and "feels like" it should be in "probabilistic polynomial time".

To circumvent this, we introduce a "promise" — namely, any algorithm attempting to decide A is "promised" that M is a valid BPP machine, and if this promise is "broken" (i.e. M is not a valid BPP machine), then A is allowed to act arbitrarily. We may formalize this as follows. Recall a *language* $L \in \{0,1\}^*$ partitions $\{0,1\}^*$ into two sets: L_{yes} (YES instances) and L_{no} (NO instances).

Definition 2 (Promise problem)**.** *A* promise problem $\mathbb{A}$ *is a partition of* $\{0,1\}^*$ *into* three *sets:* A_{yes} *(YES instances),* A_{no} *(NO instances),* A_{inv} *(invalid instances).*

Definition 3 (PromiseBPP)**.** *A promise problem* $\mathbb{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ *is in PromiseBPP if there exists a (deterministic) TM* M *and fixed polynomials* $s_A, r_A : \mathbb{N} \mapsto \mathbb{R}^+$*, such that for any input* $x \in \{0,1\}^n$*,* M *takes in string* $y \in \{0,1\}^{s_A(n)}$*, halts in at most* $O(r_A(n))$ *steps, and:*

- *(Completeness/YES case) If* $x \in A_{\text{yes}}$*, then for at least* $2/3$ *of the choices of* $y \in \{0,1\}^{s_A(n)}$*,* M *accepts.*
- *(Soundness/NO case) If* $x \in A_{\text{no}}$*, then for at most* $1/3$ *of the choices of* $y \in \{0,1\}^{s_A(n)}$*,* M *accepts.*
- *(Invalid case) If* $x \in A_{\text{inv}}$*, then* M *may accept or reject arbitrarily.*

Exercise. What are the differences between the definitions of BPP and PromiseBPP?

Exercise. Why does the naive strategy for deciding L succeed in placing $\mathbb{A} \in$ PromiseBPP?

Why all the fuss? There is a reason we are clearly delineating BPP from PromiseBPP here. The classical complexity theory community correctly distinguishes between BPP and PromiseBPP, as both are meaningful classes classically. The quantum complexity community, on the other hand, has a nasty habit of using terminology BQP to really mean PromiseBQP — this is not entirely surprising, as essentially any problem we care about quantumly is a promise problem. In an act of preserving sanity, the community has thus quietly decided not to carry around the "Promise" moniker each time BQP is mentioned. Let us hence clearly state the following, and promptly forget about it:

Throughout these notes, BQP shall be taken to mean PromiseBQP.

This distinction may seem petty, but it is actually crucial. For example[3], whereas BPP does *not* have known complete problems (due to its semantic definition), we shall see in an upcoming lecture that BQP (i.e. PromiseBQP) *does* have complete problems.

2 BQP

BQP is the natural quantum generalization of PromiseBPP: It is the set of promise problems which can be solved by a poly-time quantum computation with bounded error. To make this formal, we first make a switch from Turing machines to circuits.

[3]Another illustration of this distinction is that BPP does not have time hierarchy theorems, but PromiseBPP does. Here, a time hierarchy theorem roughly says that a PromiseBPP machine running in, say, n^2 steps can solve more promise problems than a PromiseBPP machine running in only n steps. Intuitively, a time hierarchy theorem is exactly what one would expect to hold for a reasonable model of computation, so in this sense PromiseBPP seems more natural than BPP.

From Turing machines to circuits. Although quantum Turing machines are well-defined, it turns out that quantumly, the most natural computational model to work with is the *circuit model* (i.e. applying gates like Pauli matrices to wires, as done in Lecture 2). Unlike a TM, however, which can take strings $x \in \{0,1\}^*$ of arbitrary size as input, a circuit's size (by which we mean number of wires here) is *fixed*. Thus, given an input $x \in \{0,1\}^*$, we require an efficient algorithm for generating a description of a circuit C which takes inputs of the right size, $|x|$.

Definition 4 (P-uniform quantum circuit family)**.** *A family of quantum circuits $\{Q_n\}$ is called P-uniform if there exists a polynomial-time TM M, which given as input 1^n, outputs a classical description of Q_n.*

Exercise. Why does M take in 1^n as input, as opposed to n in binary as input?

We can now define BQP (which again, really means PromiseBQP).

Definition 5 (Bounded-error quantum polynomial-time (BQP))**.** *A promise problem $\mathbb{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ is in BQP if there exists a P-uniform quantum circuit family $\{Q_n\}$ and polynomial $q : \mathbb{N} \mapsto \mathbb{N}$ satisfying the following properties. For any input $x \in \{0,1\}^n$, Q_n takes in $n + q(n)$ qubits as input, consisting of the input x in register A and $q(n)$ ancilla qubits initialized to $|0\rangle$ in register B. The first qubit of B, denoted B_1, is the designated output qubit, a measurement of which in the standard basis after applying Q_n yields the following:*

- *(Completeness/YES case) If $x \in A_{\text{yes}}$, then Q_n accepts with probability at least $2/3$.*
- *(Soundness/NO case) If $x \in A_{\text{no}}$, then Q_n accepts with probability at most $1/3$.*
- *(Invalid case) If $x \in A_{\text{inv}}$, Q_n may accept or reject arbitrarily.*

Exercise. Observe that a key difference between PromiseBPP and BQP is that the former can be viewed as a deterministic TM taking in a "random string" as input. In other words, the randomness could be "decoupled" from the TM. Do you think the randomness inherent in quantum cicuits could be "decoupled" from the quantum circuit Q_n in BQP?

Error reduction. Note that the completeness/soundness parameters $2/3$ and $1/3$ are not special (for both PromiseBPP and BQP). By simply repeating the circuit Q_n polynomially many times in parallel, measuring each output qubit, and then accepting if and only if a majority of the runs output 1, we may improve the error parameters to $1 - 2^n$ for completeness and 2^n for soundness.

Exercise. The error reduction claim above can be proven formally via the following *Chernoff bound*: If $X_1, \ldots, X_m$ are identically and independently distributed random variables over $\{0,1\}$, such that each X_i has value 1 with probability $1/2 + \epsilon$, then the probability that $\sum_{i=1}^m X_i$ is strictly smaller than $m/2$ is at most $e^{-2\epsilon^2 m}$. With this bound in hand, how many repetitions m of a BQP circuit Q_n are required to get the completness to at least $1 - 2^{-n}$?

2.1 Norms and perturbations to quantum gates

We will frequently need to measure distances between states and gates in this course; a crucial example of this will be in Section 2.2 below on universal gate sets. For this, we need to broaden our repertoire of norms.

Norms on matrices. Just as the Euclidean norm $\| |\psi\rangle \|_2$ gives a notion of "size" for vector $|\psi\rangle$, one may define norms on matrices. The two most common ones in quantum information are:

- **Spectral Norm:** The spectral norm of operator $M \in \mathcal{L}(\mathbb{C}^d)$ is

$$\| M \|_\infty := \max_{\text{unit vectors } |\psi\rangle \in \mathbb{C}^d} \| M|\psi\rangle \|_2 .$$

  Thus, $\| M \|_\infty$ measures the maximum amount M can "stretch" some vector in $\mathbb{C}^d$.

Exercise. Define $M = 3|0\rangle\langle 0|$. What is $\| M \|_\infty$? Visualize the action of M in the 2D real Euclidean plane to intuitively confirm your calculation.

- **Trace Norm:** The trace norm of operator $M \in \mathcal{L}(\mathbb{C}^d)$ is

$$\| M \|_{\text{tr}} := \text{Tr}(|M|) = \text{Tr}\sqrt{M^\dagger M},$$

where note $|M|$ and $\sqrt{M^\dagger M}$ are operator functions.

Exercise. Let M be Hermitian with eigenvalues λ_i. Prove that $\| M \|_{\text{tr}} = \sum_i |\lambda_i|$.

Exercise. For unit vectors $|\psi\rangle, |\phi\rangle \in \mathbb{C}^d$, prove that $\| |\psi\rangle\langle\psi| - |\phi\rangle\langle\phi| \|_{\text{tr}} = 2\sqrt{1 - |\langle\psi|\phi\rangle|^2}$. Hint: Observe that $M = |\psi\rangle\langle\psi| - |\phi\rangle\langle\phi|$ is rank at most 2. Combined with the fact that $\text{Tr}(M) = 0$, this should reveal something about the structure of the eigenvalues of M. Finally, consider what the eigenvalues of $\text{Tr}(M^2)$ look like to complete the picture.

By definition of a *norm* (which should be thought of as a generalization of the absolute value function to higher dimensions), both the spectral and trace norms satisfy (1) the triangle inequality ($\| A + B \|_\infty \leq \| A \|_\infty + \| B \|_\infty$), (2) absolute homogeneity ($\| aM \|_\infty = |a| \| M \|_\infty$ for $a \in \mathbb{C}$), and (3) positive definiteness (if $\| M \|_\infty = 0$ then $M = 0$). However, they also satisfy three powerful properties:

- The Hölder inequality, which for the spectral and trace norms yields $\left|\text{Tr}(A^\dagger B)\right| \leq \| A \|_\infty \| B \|_{\text{tr}}$.
- Submultiplicativity, which says $\| AB \|_\infty \leq \| A \|_\infty \| B \|_\infty$.
- Invariance under unitaries, i.e. for any unitaries U and V, $\| UMV \|_\infty = \| M \|_\infty$.[4]

Motivating the trace norm. The trace norm may seem an odd quantity, but it has a remarkably important operational interpretation. Let us play a game: I have two density operators in mind, $\rho, \sigma \in \mathcal{L}(\mathbb{C}^d)$, of which you know a complete classical description. I privately flip a fair coin — if I get HEADS (respectively, TAILS), I prepare ρ (respectively, σ) as a quantum state and send it to you. Your task is to guess, using *any* measurement allowed by quantum mechanics (efficiently implementable or not), whether I sent you ρ or σ. Intuitively, if ρ and σ are "close" to one another, this should be "hard", and if they are "far", then, it should be "easy". It turns out that the right way to measure "closeness" is the trace norm, in that your *optimal* probability of winning this game is precisely

$$\frac{1}{2} + \frac{1}{4} \| \rho - \sigma \|_{\text{tr}}.$$

This is a deep statement worth reflecting on — the trace norm characterizes how *distinguishable* two quantum states are.

Exercise. What is your probability of winning the game above if $\rho = \sigma$? How about if $\rho = |0\rangle\langle 0|$ and $\sigma = |1\rangle\langle 1|$?

[4] There is an elegant reason why these norms satisfy invariance under unitaries — the norms are really "classical" norms in disguise, in that they do not depend on the choice of basis. Define for vector $|\psi\rangle \in \mathbb{C}^d$ the Schatten p-norm $\| |\psi\rangle \|_p = (\sum_{i=1}^d |\psi_i|^p)^{1/p}$, with $\| |\psi\rangle \|_\infty = \max_{i=1}^d |\psi_i|$. (What norm is recovered for $p = 2$?) Recall that every matrix $M \in \mathbb{C}^{d_1 \times d_2}$ can be written with respect to its *singular value decomposition* $M = \sum_{i=1}^{\min d_1, d_2} s_i |l_i\rangle\langle r_i|$, for singular values $s_i \in \mathbb{R}^+$, orthonormal left singular vectors $\{|l_i\rangle\} \subseteq \mathbb{C}^{d_1}$, and orthonormal right singular vectors $\{|r_i\rangle\} \subseteq \mathbb{C}^{d_2}$. (This should remind you of the Schmidt decomposition, which is no coincidence.) Then, the spectral and trace norms of A are just the ∞- and 1-norms applied to the vector of singular values of A, respectively. But since unitaries U, V applied as UMV leave the singular values invariant, it follows that these norms are unitarily invariant.

Perturbations to quantum gate sequences. With norms in hand, we can study how errors affect quantum computations. Suppose we have a quantum circuit $U = U_m \cdots U_2 U_1$ for unitary operators U_i in mind, but experimentally we are not able to implement each U_i perfectly. Instead, for each i, we can implement some unitary U_i' satisfying $\| U_i - U_i' \| \leq \epsilon$ for $\| \cdot \|$ a unitarily invariant norm. The following useful lemma tells us how this per-error gate accumulates as we apply each U_i.

Lemma 6. *Let $\| \cdot \|$ be a norm satisfying unitary invariance. Let $U = U_m \cdots U_2 U_1$ and $U' = U_m' \cdots U_2' U_1'$ be a pair of quantum circuits, for unitaries U_i, U_i' satisfying $\| U_i - U_i' \| \leq \epsilon$ for all $i \in [m]$. Then, $\| U - U' \| \leq m\epsilon$.*

Proof. We proceed by induction on m. The base case $m = 1$ is trivial. Assume the claim holds for $m > 1$, and for brevity, let $V = U_{m-1} \cdots U_1$ and $V' = U_{m-1}' \cdots U_1'$. Then,

$$\begin{aligned} \| U - U' \| &= \| U_m V - U_m' V' + (U_m V' - U_m V') \| \\ &= \| U_m (V - V') + (U_m' - U_m) V' \| \\ &\leq \| U_m (V - V') \| + \| (U_m' - U_m) V' \| \\ &= \| V - V' \| + \| U_m' - U_m \| \\ &\leq (m-1)\epsilon + \epsilon, \end{aligned}$$

where the first statement follows by adding and subtracting $U_m V'$, the third by the triangle inequality, the fourth by unitary invariance, and the fifth by the induction hypothesis. □

Thus, the error propagates nicely (i.e. linearly). As far as BQP is concerned, if we make an exponentially small additive error per gate, the overall error will hence be negligible (since BQP circuits contain polynomially many gates). Indeed, we can even get by with sufficiently small inverse polynomial additive error per gate.

Exercise. How can the requirement that the norm be unitarily invariant in Lemma 6 be relaxed? (Hint: Which of the other properties of the trace and spectral norms would also suffice?

How measurement statistics are affected by errors in quantum circuits. We have established how gate-wise error propagates through a circuit. Now comes the real question — as an experimenter in the lab looking at actual measurement statistics, what do the error bounds of Lemma 6 mean? For this, we state the following lemma, whose proof is the subsequent exercise.

Lemma 7. *Let $\rho \in \mathcal{D}(\mathbb{C}^d)$ be a quantum state, $\Pi \in \mathrm{Pos}(\mathbb{C}^d)$ a projector, and $U, V \in \mathcal{U}(\mathbb{C}^d)$ unitaries such that $\| U - V \|_{\mathrm{tr}} \leq \epsilon$. Then,*

$$\left| \mathrm{Tr}(\Pi U \rho U^\dagger) - \mathrm{Tr}(\Pi V \rho V^\dagger) \right| \leq 2\epsilon.$$

In words, any projective measurement outcome Π's probability changes by at most 2ϵ when evolving under V instead of U.

Exercise. Prove Lemma 7 using an approach similar to the proof of Lemma 6. (Hint: Begin by writing the left hand side as $\left|\mathrm{Tr}(\Pi(U\rho U^\dagger - V\rho V^\dagger))\right|$ and applying a useful inequality.)

2.2 Universal quantum gate sets

With Lemma 7 in hand, we can now discuss the fundamental notion of a *universal quantum gate set*. As computer scientists, in this course we shall measure the "cost" of a circuit by the number of gates it contains. From an experimental standpoint, however, since the set $\mathcal{U}(\mathbb{C}^d)$ is uncountably infinite, this seems somewhat unrealistic — implementing an arbitrary element of an uncountable set on demand is a tall order. Instead, we follow the classical route and recall that the NOT and AND gates together are *universal* for classical computation; in other words, any classical circuit can be rewritten using NOT and AND.

Amazingly, a similar statement holds quantumly — there exist finite universal gate sets $S \subseteq \mathcal{U}(\mathbb{C}^d)$, one of which is $S = \{H, \pi/8, CNOT\}$. Here,

$$\pi/8 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

S is universal in the intuive sense that any unitary $U \in \mathcal{U}(\mathbb{C}^d)$ can be approximated with gates from S. The question is: *How good is the approximation, and what overhead does it incur?*

There are two parts to the answer. First, an arbitrary unitary $U \in \mathcal{U}((\mathbb{C}^2)^{\otimes n})$ can be decomposed *exactly* as a product of CNOT gates and arbitrary 1-qubit gates. However, this decomposition is *not* necessarily efficient, requiring $O(n^2 4^n)$ gates in the worst case. But at least we are reduced to needing only a single 2-qubit gate, the CNOT, and now we need to approximate arbitrary 1-qubit gates. For this, the Solovay-Kitaev theorem says that an arbitrary 1-qubit gate can be approximated within ϵ additive error (with respect to, say, trace or spectral norm) using $O(\log^c(1/\epsilon))$ gates from $\{H, \pi/8\}$ for $c \approx 2$.

For the purposes of BQP, the overhead incurred by the Solovay-Kitaev theorem is very good — we may get sufficiently small inverse polynomial additive error with only a polylogarithmic overhead in gate count. As a result, BQP takes on a particularly well-motivated form: Without loss of generality, we may assume in Definition 5 that the Turing machine generating the P-uniform quantum circuit family only needs to pick gates from the universal set $S = \{H, \pi/8, CNOT\}$.

2.3 The BQP subroutine problem

The error reduction of BQP turns out to have a "boring" but crucial implication — if we have a BQP circuit solving a particular promise problem A, then we can use that circuit as a *subroutine* in solving other problems.

Classically, the analogous statement is trivial, as you know from the many times you have likely called (say) methods in Java as subroutines. So why is it non-trivial quantumly? Here is a sketch: Suppose we have a quantum circuit C, which applied to state $|x\rangle|0^m\rangle$ (for $x \in \{0,1\}^n$ the input) produces state $|\psi\rangle$ on $n+m$ qubits. If C accepts with probability exactly $2/3$ (i.e. $x \in A_{\text{yes}}$), then we may write

$$C|x\rangle|0^m\rangle = |\psi\rangle = \sqrt{\frac{1}{3}}|0\rangle_1|\psi_0'\rangle_{2,\ldots,n+m} + \sqrt{\frac{2}{3}}|1\rangle_1|\psi_1'\rangle_{2,\ldots,n+m}$$

for some $|\psi_0'\rangle, |\psi_1'\rangle \in (\mathbb{C}^2)^{\otimes(n+m-1)}$. In other words, the output qubit is potentially highly entangled with the remaining $n+m-1$ qubits (which happens, if say $|\psi_0'\rangle$ and $|\psi_1'\rangle$ are orthogonal). Since we are treating C as a subroutine, presumably we will only act in the future on this output qubit, and not the remaining $n+m-1$ qubits output by C. But recall the principle of deferred measurement states that we might as well thus assume that the remaining $2, \ldots, n+m-1$ have been discarded or *traced out*. Thus, if the output qubit is indeed entangled with the remaining qubits, we are in trouble — our machine will effectively be in a highly mixed state after we make the subroutine call! This is not at all what we were expecting. However, if we first reduce the error of C via parallel repetition, we know that we instead have (say)

$$C|x\rangle|0^m\rangle = |\psi\rangle = \sqrt{\frac{1}{2^n}}|0\rangle_1|\psi_0'\rangle_{2,\ldots,n+m} + \sqrt{1-\frac{1}{2^n}}|1\rangle_1|\psi_1'\rangle_{2,\ldots,n+m}.$$

While qubit 1 can still be entangled with qubits $2, \ldots n+m$ in this case, note that the vast majority of the amplitude is on the $|1\rangle_1$ branch — so tracing out qubits $2, \ldots, m+n$ will have only exponentially small disturbance on our state. Since our computation takes only polynomially many steps and is bounded error, such an exponentially small error at each step is negligible.

Exercise. One way to formalize the idea that the error is negligible is as follows. Define unit vectors:

$$\begin{aligned} |\psi\rangle &= \alpha|0\rangle_1|\psi_0'\rangle_{2,\ldots,n+m} + \beta|1\rangle_1|\psi_1'\rangle_{2,\ldots,n+m}, \\ |\phi\rangle &= |1\rangle_1|\psi_1'\rangle_{2,\ldots,n+m}. \end{aligned}$$

Prove that $\| |\psi\rangle - |\phi\rangle \|_2 = \sqrt{2}\sqrt{1 - \mathrm{Re}(\beta)}$, for $\mathrm{Re}(\beta)$ the real part of β.

Exercise. Suppose $\| |\psi\rangle - |\phi\rangle \|_2 \leq \epsilon$. For any measurement projector Π, how can you upper bound the quantity $|\mathrm{Tr}(\Pi|\psi\rangle\langle\psi|) - \mathrm{Tr}(\Pi|\phi\rangle\langle\phi|)|$ in terms of ϵ? The precise formula is slightly messy, so it suffices to sketch which inequalities/equalities to chain together to derive a bound.

3 Relationship to other complexity classes

Given that quantum computers appear to outperform classical ones when it comes to the factoring problem, a natural question arises: Just how powerful are quantum polynomial-time computations? With BQP acting as a reasonable formal definition of what we view as "efficient quantum computation", we are in a position to make progress on this question.

Containment in PSPACE. The naive hope is that since quantum computers can put n-qubit systems into 2^n states simultaneously, perhaps one can simulate exponential-time classical computations in BQP. In other words, perhaps EXP $\subseteq$ BQP, for EXP the analogue of P in which the Turing machine may take exponentially many steps. This hope is dashed rather easily, however, via the following preliminary theorem, whose simple but elegant proof using a notion of "Feynman path integrals" repeatedly finds applications in the field.

Theorem 8. BQP $\subseteq$ PSPACE.

Recall that PSPACE is the set of decision problems decidable by a polynomial-*space* Turing machine, formally defined as follows.

Definition 9 (Polynomial-Space (PSPACE))**.** *A language $L \subseteq \{0,1\}^*$ is in* PSPACE *if there exists a (deterministic) TM M and fixed polynomial $s_L : \mathbb{N} \mapsto \mathbb{R}^+$, such that for any input $x \in \{0,1\}^n$, M uses at most $O(s_L(n))$ cells on its work tape, and:*

- *(Completeness/YES case) If $x \in L$, M accepts.*
- *(Soundness/NO case) If $x \notin L$, M rejects.*

Exercise. How many steps can a PSPACE algorithm take in the worst case?

Exercise. Show that BQP $\subseteq$ EXP.

Proof of Theorem 8. Let x be an instance of any BQP promise problem $\mathbb{A} = (A_{\mathrm{yes}}, A_{\mathrm{no}}, A_{\mathrm{inv}})$ of length n. Then, there exists a poly-time TM M which given $|x|$ as input, outputs a quantum circuit $Q_n = U_m \cdots U_1$ of 1- and 2-qubit gates. Measuring the output qubit of M in the standard basis outputs 1 with probability at least 2/3 if $x \in A_{\mathrm{yes}}$, or outputs 1 with probability at most 1/3 if $x \in A_{\mathrm{no}}$. Thus, it suffices to sketch a PSPACE computation which estimates the probability of outputting 1 to within additive error, say, 1/12.

Let $\Pi_1 = |1\rangle\langle 1| \in \mathcal{L}(\mathbb{C}^2)$ be a projective measurement onto the output qubit of Q_n, and $|\psi\rangle = Q_n|x\rangle|0^{q(n)}\rangle$. The probability of outputting 1 is

$$\Pr[\text{output } 1] = \mathrm{Tr}(\Pi_1|\psi\rangle\langle\psi|) = \langle\psi|\Pi_1|\psi\rangle.$$

Remark. *Note that $|\psi\rangle$ is an $n + q(n)$ qubit state, but Π_1 acts only on one qubit. To ensure dimensions match, throughout this course we implicitly mean $\Pi_1 \otimes I_{2,\ldots,n+q(n)}$ in such cases. This implies the probability of outputting* 1 *can equivalently be written*

$$\Pr[\textit{output } 1] = \mathrm{Tr}(\Pi_1 \cdot \mathrm{Tr}_{2,\ldots,n+q(n)}(|\psi\rangle\langle\psi|)).$$

Expanding the definition of $|\psi\rangle$,

$$\Pr[\text{output } 1] \quad = \quad \langle x|\langle 0^{q(n)}|U_m^\dagger \cdots U_1^\dagger \Pi_1 U_m \cdots U_1|x\rangle|0^{q(n)}\rangle.$$

Now, let us do something seemingly trivial — insert an identity operator I between each pair of operators, and use the fact that for N-qubit I, we may write $I = \sum_{x \in \{0,1\}^N} |x\rangle\langle x|$ (i.e. I diagonalizes in the standard basis with eigenvalues 1). Then:

$$\begin{aligned}\Pr[\text{output 1}] &= \langle x|\langle 0^{q(n)}| \cdot I \cdot U_m^\dagger \cdot I \cdots I \cdot U_1^\dagger \cdot I \cdot \Pi_1 \cdot I \cdot U_m \cdot I \cdots I \cdot U_1 \cdot I \cdot |x\rangle|0^{q(n)}\rangle \\ &= \sum_{x_1,\ldots,x_{2m+2} \in \{0,1\}^{n+q(n)}} (\langle x|\langle 0^{q(n)}|)|x_1\rangle\langle x_1|U_m^\dagger \cdot |x_2\rangle \cdots \langle x_{2m+1}|U_1|x_{2m+2}\rangle\langle x_{2m+2}|(|x\rangle|0^{q(n)}\rangle).\end{aligned}$$

For each fixed $x_1, \ldots, x_{m+2}$, note the summand is a product of $2m+3$ complex numbers of the form (for example) $\langle x_1|U_m^\dagger|x_2\rangle$. Since each U_i is at most a 2-qubit unitary, and since $\langle x_i|U|x_j\rangle$ is just the entry of U at row x_i and column x_j, we may compute this product in polynomial time.

Exercise. Convince yourself that one may compute $\langle x|I_{1,\ldots,n-1} \otimes U_n|y\rangle$ in polynomial time, for $x, y \in \{0,1\}^n$ and U a 2-qubit unitary acting on qubit n.

Exercise. There is a subtlety in the previous exercise we must be wary of: If each term in the summand is specified to (say) $O(1)$ bits, then how many bits can the product of all $2m+3$ terms have? Does this contradict your argument that the multiplication can be done in polynomial time? What if each term in the summand is specified to $O(n)$ bits?

Returning to the proof, we are essentially done — for each summand takes polynomially many steps on a TM to compute (and hence polynomial space), and the number of summands is $(2^{n+q(n)})^{2m+2}$, where by definition of BQP, m is polynomially bounded in n. Thus, although the number of summands is exponential in n, we can simply use a counter (requiring $\lceil \log((2^{n+q(n)})^{2m+2}) \rceil$ bits, i.e. polynomially many bits) to iterate through all summands one by one, each time adding the current summand to a running total. The final value of the running total is, by definition, the acceptance probability of circuit Q_n, as desired.

Exercise. How does the addition of all summands grow the number of bits required to represent the answer? In other words, given N n-bit rational numbers z_i, what is an upper bound on the number of bits required to represent $\sum_i z_i$?

Remark. *We noted at the outset of the proof that it would suffice to compute the acceptance probability of Q_n to within, say, $1/12$ additive error. So how accurate was our polynomial-space summation technique? This depends on how many bits of precision are used to specify the entries of each unitary U_i. If the gates U_i can be specified* exactly *using* poly(n) *bits, then the summation is* perfect*. If, however, we have irrational entries in our gates, such as a $1/\sqrt{2}$ for the Hadamard, one approach is to approximate each entry up to $p(n)$ bits for p a sufficiently large polynomial. In other words, we can approximate each entry up to an additive error of $2^{-p(n)}$. One can then show that for sufficiently large but fixed p, the overall additive error over the entire summation can be made exponentially small in n, allowing us to distinguish YES from NO cases, as desired.* □

Better upper bounds. While the technique of Theorem 8 is elegant, PSPACE is not the best known upper bound on BQP. One can place BQP in the presumably smaller, but still powerful, class PP, which is the unbounded-error analogue of BPP.

Theorem 10. BQP $\subseteq$ PP.

Since we will discuss PP in the future, let us formally define it for completeness.

Definition 11 (Probabilistic Polynomial Time (PP))**.** *A language $L \subseteq \{0,1\}^*$ is in PP if there exists a (deterministic) TM M and fixed polynomials $s_L, r_L : \mathbb{N} \mapsto \mathbb{R}^+$, such that for any input $x \in \{0,1\}^n$, M takes in string $y \in \{0,1\}^{s_L(n)}$, halts in at most $O(r_L(n))$ steps, and:*

- *(Completeness/YES case) If $x \in L$, then for strictly larger than $1/2$ of the choices of $y \in \{0,1\}^{p_L(n)}$, M accepts.*
- *(Soundness/NO case) If $x \notin L$, then for at most $1/2$ of the choices of $y \in \{0,1\}^{p_L(n)}$, M rejects.*

We say M accepts *(*rejects*) x in the YES case (NO case).*

There are various ways to prove Theorem 10. The approach we will take in an upcoming lecture is to actually prove that Quantum-Merlin Arthur (QMA), a quantum analogue of NP which trivially contains BQP, is itself contained in PP; this will be shown via *strong error-reduction* for QMA. Two other approaches use completely different techniques: The first is to show that PostBQP, which trivially contains BQP and is roughly BQP with the magical ability to "postselect" on certain outputs, equals PP. The second is to use a technique known as *hierarchical voting* to show that $\mathrm{P}^{\mathrm{QMA[log]}}$, which is the class of decision problems decidable by a P machine making logarithmically many queries to a QMA oracle (which hence trivially contains QMA, and thus also BQP), is also contained in PP.

Finally, let us close by remarking that, in addition to $\mathrm{P}^{\mathrm{QMA[log]}}$, another upper bound on QMA stronger than PP is known: $\mathrm{BQP} \subseteq \mathrm{QMA} \subseteq \mathrm{A_0PP} \subseteq \mathrm{PP}$. And while $\mathrm{A_0PP}$ is a rather strange class, it does have the nice property that if $\mathrm{A_0PP} = \mathrm{PP}$, then PH collapses (which is highly unlikely). Thus, as far as theoretical computer science is concerned, QMA (and hence BQP) are *strictly* contained in PP.