Quantum Complexity Theory Sevag Gharibian
Summer 2019, University of Paderborn

Lecture 10: BQP versus the Polynomial Hierarchy

"And it is also said," answered Frodo: "Go not to the Elves for counsel for they will answer both no and yes." "Is it indeed?" laughed Gildor. "Elves seldom give unguarded advice, for advice is a dangerous gift, even from the wise to the wise, and all courses may run ill."
— J. R. R. Tolkien, The Fellowship of the Ring

Contents

Introduction. In Assignment 3, we showed the Sipser-Gács-Lautemann Theorem, which stated that BPP $\subseteq$ PH (more precisely, BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$). A natural question is thus: Could BQP $\subseteq$ PH as well?

At first glance, this question appears wholly unconnected to our study of Boson Sampling from Lecture 9. A closer look, however, reveals similar magic at play: BPP $\subseteq$ PH is shown by leveraging the fact that randomness can be "extracted" from a randomized Turing machine; in this sense, a randomized Turing machine can be viewed as deterministic, instead taking a uniformly random string as input. Whether the inherent randomness in *quantum* circuits can similarly be extracted, however, is entirely unclear. Indeed, it was precisely this distinction between the classical and quantum models which was exploited in Boson Sampling to argue that if a *classical* (but not quantum!) algorithm could solve the approximate Boson Sampling problem, then PH is at risk[1] of collapsing. It hence seems this distinction between "classical and quantum randomness" has an important role to play in delineating the power of classical versus quantum computation. Indeed, for this reason, the techniques of the Sipser-Gács-Lautemann Theorem break down in the quantum setting, and BQP is generally believed *not* to lie in PH, in contrast to BPP.

Of course, proving BQP $\not\subseteq$ PH is difficult. In this lecture, we discuss arguably the next best thing: "Evidence" that BQP is not in PH in the form of an *oracle separation* between the classes. The word evidence is in quotes here, as recall oracle separations are *not* necessarily reliable evidence that a pair of classes are distinct. For example, there exist oracles A and B relative to which one can rigorously prove $\mathrm{P}^A = \mathrm{NP}^A$ and yet $\mathrm{P}^B \neq \mathrm{NP}^B$. This is the meaning of the opening quote of this lecture — oracle separations are like Elves; they may "answer" both no and yes, and it is not clear what the correct answer should be. (As an aside, what oracle separations *do* rigorously show is that any separation proof between (in this case) P and NP must be non-relativizing, i.e. must break down when oracles are added to the picture.) Nevertheless, as separating classes is typically difficult, oracle separations are often viewed as a desirable first step in this direction.

[1] We say "at risk" of collapsing, as recall in the setting of approximate Boson Sampling, part of the research agenda currently relies on conjectures.

Organization. We begin in Section 1 by stating and parsing the key claim on which the lecture rests. This includes fleshing out a connection between bounded depth circuits and alternating quantifiers in Section 1.1. The remainder of the lecture shows the key claim: Section 2 states the distribution D required for the claim, and Sections 3 and 4 sketch its proof. This lecture uses some useful tools such as Fourier analysis to study Boolean functions, which are worth delving into in their own right. A nice reference for the latter is `https://arxiv.org/abs/1205.0314` (Analysis of Boolean Functions, lecture notes of Ryan O'Donnell with a guest lecture by Per Austrin, scribed by Li-Yang Tan).

1 The key claim

We begin by stating and parsing the key claim on which the lecture rests. Throughout this lecture, we set $N := 2^n$ for n the input parameter of interest (i.e. N is exponentially large).

Theorem 1. *Let U_{2N} denote the uniform distribution over $\{\pm 1\}^{2N}$. There exists an explicit distribution D over $\{\pm 1\}^{2N}$ satisfying both of the following:*

1. *(Distinguishing is easy quantumly) There exists a quantum algorithm able to distinguish U_{2N} from D*
   - *with advantage $\Omega(1/\log N)$, and*
   - *in time $O(\log N)$ using 1 input query.*
2. *(Distinguishing is hard classically) Any Boolean circuit of size quasipolynomial in N and of constant depth cannot distinguish U_{2N} from D with advantage better than $O(\text{polylog}(N)/\sqrt{N})$.*

Exercise. An important step in digesting technical material is to understand what you don't understand — which terms in Theorem 1 above require clarification in order to make the theorem a formal statement?

As suggested by the exercise above, there are a few terms here which require clarification:

- *What is a Boolean circuit in this context?* By "Boolean circuit", we mean a classical circuit consisting of unbounded fan-in[2] AND and OR gates, as well as NOT gates. The circuit has a single output wire. Its *size* is the number of gates, and its *depth* is the length of the longest path in edges from any input wire to the output wire. (Here, we are implicitly viewing the circuit as a directed acyclic graph from input wires to output wires.)
- *What do we mean by "advantage"?* Let D, D' be distributions over a finite set X. Then, we say a (classical or quantum) algorithm A distinguishes between D and D' with advantage ϵ if

$$|\Pr_{X\sim D}[A \text{ accepts } x] - \Pr_{X\sim D'}[A \text{ accepts } x]| = \epsilon.$$

- *How are the distributions U_{2N} and D accessed?* We stated above that N is exponentially large in n; thus, each random sample $x \in \{\pm 1\}^{2N}$ is an exponentially large string. To make this a meaningful input model, we hence grant algorithms *oracle access* to string x:
  - A classical algorithm is allowed to perform the mapping $i \mapsto x_i$ for unit cost, for $i \in [\log(2N)]$ and x_i the ith bit (in the ± 1 basis) of x.
  - A quantum algorithm is allowed to coherently perform the mapping $|i\rangle \mapsto x_i|i\rangle$ for unit cost. Here, we are implicitly using phase kickback to inject $x_i \in \{\pm 1\}$ as a phase. The mapping also works with any "garbage" in an ancilla register, i.e. $|i\rangle|g\rangle \mapsto x_i|i\rangle|g\rangle$ for any state $|g\rangle$.
- *What counts as "quasipolynomial" size in N?* Typically, quasipolynomial in N refers to quantities such as $O(2^{\log^c N})$ for constant c.

[2] By unbounded fan-in, we mean each AND and OR gate can have multiple input wires, as opposed to just 2.

- *What in the world does Theorem 1 have to do with* PH*?* As the title of this lecture suggests, we are interested in giving an oracle-based task which can be solved in BQP but not in PH. Yet, the statement of Theorem 1 says nothing about PH or even alternating quantifiers — rather, it is a no-go statement for quasipolynomial size bounded depth circuits. The missing link between these two ideas is worthy of a discussion in its own right, which we now move to.

1.1 The connection between bounded depth circuits and alternating quantifiers

We first recall the definition of PH.

Definition 2 (Polynomial Hierarchy (PH) and Σ_k^p)**.** *A language $L \subseteq \{0,1\}^*$ is in Σ_k^p, if there exists a polynomial-time uniformly generated Turing machine M which takes in input $x \in \{0,1\}^n$, k proofs $y_1,\ldots,y_k \in \{0,1\}^{n^c}$ for $c,k \in O(1)$, and acts as follows:*

$$x \in L \quad \Rightarrow \quad \exists y_1 \forall y_2 \exists y_3 \cdots Q_k y_k \text{ such that } M \text{ accepts } (x, y_1, \ldots, y_k), \tag{1}$$

$$x \notin L \quad \Rightarrow \quad \forall y_1 \exists y_2 \forall y_3 \cdots \overline{Q}_k y_k \text{ such that } M \text{ rejects } (x, y_1, \ldots, y_k), \tag{2}$$

where $Q_k = \exists$ ($Q_k = \forall$) if i is odd (even). We define $\mathrm{PH} = \bigcup_{i \in \mathbb{N}} \Sigma_k^p$*, where* $\Sigma_0^p = \mathrm{P}$*.*

In the setting of *oracle* problems, there is a slick connection between bounded depth circuits and computations of the above form, which we state and prove in full generality below. To set this up, fix $N = 2^n$ for input parameter n, and consider an arbitrary Boolean function $f : \{0,1\}^N \mapsto \{0,1\}$. Since any input $x \in \{0,1\}^N$ to f is exponentially long, we assume we are given only access via some oracle O_x to x, i.e. the ability to map $i \mapsto x_i$ for unit cost for any $i \in [N]$.

Lemma 3. *Suppose there exists a Σ_k^p machine M which is given as input (1) a unary string 1^n and (2) $x \in \{0,1\}^N$ given implicitly via access to oracle O_x, and computes output $f(x) \in \{0,1\}$. Then, there exists a Boolean circuit of depth $k+1$ and size $O(2^{\log^{c'} N})$ for some $c' \in O(1)$ which, given access to oracle O_x, computes $f(x)$.*

It is worth stressing above that M only receives 1^n as an explicit input; the "actual" input x on which f is to be evaluated is "stored off-site" in the oracle O_x.

Proof of Lemma 3. Let M be the Σ_k^p machine computing $f(x)$ with oracle access to $x \in \{0,1\}^N$. Without loss of generality, we may assume M makes only a single query to x, as per the following exercise.

Exercise. Show that by adding two additional alternating quantifiers, one can simulate M making a polynomial number of calls to f with some M' making only a single call to f. (Hint: Think about computational paths which branch each time an oracle query answer bit is received. Use the $\exists$ and $\forall$ quantifiers to "pick out and enforce" the "correct" computational branching process.) Can you reduce it to requiring just one additional alternating quantifier?

High-level idea. We shall build an AND-OR tree T whose nodes are unbounded fan-in AND and OR gates. The leaves are the input bits; by applying the AND and OR gates each time we move up a level from the leaves, we arrive at the root, which shall be an OR gate. The output bit of the root shall be 1 if and only if $f(x) = 1$. The depth of the tree shall be $k+1$, and its size $O(2^{\log^{c'} N})$; hence, we will have a circuit computing $f(x)$ with properties stated in the claim.

The construction. We view the action of M via its computational branches. Let us start with the first existentially quantified proof, $y_1 \in \{0,1\}^{n^c}$. This induces a branching in M over 2^{n^c} computational paths (one per possible proof y_1), and M accepts if at least one of these branches accepts. Hence, in T we "represent y_1" via an OR gate at the root which takes in 2^{n^c} wires and outputs a single wire. We can now recursively apply the same idea for each successive proof y_i, except whenever we have $Q_i = \forall$, we instead put in an AND gate into T (as opposed to an OR gate for $Q_i = \exists$). Finally, each leaf of T is reached by fixing a sequence of proofs $y_1, \ldots, y_k$. At any such leaf, we may assume M makes its single query to O_x, and subsequently decides to accept or reject. Specifically, M evaluates some polynomial-time function $g(y_1, \ldots, y_k) : (\{0,1\}^{n^c})^{\times k} \mapsto [N]$ to obtain the index $i \in [N]$ on which it will query O_x. Upon obtaining x_i, it performs some final polynomial-time computation $g'(y_1, \ldots, y_k, x_i) : (\{0,1\}^{n^c})^{\times k} \times \{0,1\} \mapsto \{0,1\}$ to decide whether to accept or reject. Equivalently, this can be viewed as: Conditioned on $y_1, \ldots, y_k$, M either returns x_i or $\overline{x_i}$.

Exercise. There is another option above: M could return a constant value independent of x_i. Why can we ignore this case without loss of generality? (Hint: Do we need a query to O_x in this case? If no query is needed, how can we trivially modify the tree and corresponding circuit to eliminate this leaf altogether?)

Now, if M returns x_i at this leaf, then the corresponding circuit simply reads input bit x_i here via a query to O_x. If M instead returns $\overline{x_i}$ at this leaf, the corresponding circuit first reads input x_i via O_x, and subsequently applies a NOT gate (which is also viewed as a node in T with one input and one output wire). This completes the construction.

Exercise. Prove that the tree T constructed has depth $k+1$ and size $O(\sum_{i=0}^{k}(2^{n^c})^i) \in O(2^{\log^{c'} N})$ for some $c' \in O(1)$. Conclude there is a bounded depth circuit as stated in the claim which computes $f(x)$ given oracle access to x. $\square$

With Lemma 3, we can close our discussion of the key claim of the lecture via the following exercise.

Exercise. Use Lemma 3 to answer our earlier question, "What in the world does Theorem 1 have to do with PH?". In other words, show that Theorem 1 implies that for any $k \in O(1)$, no Σ_k^p machine can distinguish between U_{2N} and D with advantage better than $O(\text{poly}(N)/\sqrt{N})$.

1.2 Outline for lecture

With Theorem 1 in place, the remainder of the lecture proceeds as follows:

- Specify the distribution D.
- Show that distinguishing D from U_{2N} is easy quantumly.
- Show that distinguishing D from U_{2N} is hard classically.

2 The distribution D

The distribution D in Theorem 1 is defined via a two-step process as follows. Set $n \in \mathbb{N}$ as our input parameter, $N = 2^n$, and $\epsilon = 1/(24\ln(N))$ (the precise value of ϵ is not relevant for our lecture, only that $\epsilon \in O(\text{poly}(n))$.

Step 1: Define a distribution G' over continuous space $\mathbb{R}^N \times \mathbb{R}^N$.

1. (Sample the first N real numbers) Sample $x_1, \ldots, X_N \in \mathbb{R}$ independently, each according to $\mathcal{N}(0,1)$. (This can roughly be viewed as choosing a Haar random vector in $\mathbb{R}^N$.) Denote $x = (x_1, \ldots, x_N) \in \mathbb{R}^N$.
2. (Correlate the second N real numbers with x) Observing that $x \in \mathbb{R}^{2^n}$ is a column vector, set $y = H^{\otimes n}x$ for H the 2×2 Hadamard gate.
3. (Final output) Output vector $\sqrt{\epsilon}(x, y) \in \mathbb{R}^{2N}$.

Note that since ϵ is "small", with high probability $-1 \leq z_i \leq 1$. For simplicity in this lecture, we henceforth assume that indeed $-1 \leq z_i \leq 1$ for all $i \in [2N]$. (One can deal with z_i violating this via a further "truncation step", which complicates the analysis and does not affect its core intuition; we omit this here.)

Step 2: "Round" G' to a distribution D over discrete space $\{\pm 1\}^N \times \{\pm 1\}^N$. The distribution G' is over $\mathbb{R}^{2N}$, but Theorem 1 requires a distribution D over $\{\pm 1\}^{2N}$. Since we are assuming all $z_i \in [-1, 1]$, we can perform such a mapping to $\{\pm 1\}^{2N}$ using a now-standard idea in approximation algorithms; namely, "snap" z_i to whichever of -1 or 1 it is closer to with probability proportional to $|z_i|$. Formally:

1. (Snap to the Boolean hypercube) Independently for each $i \in [2N]$, set
$$z_i' := \pm 1 \text{ with probability} \frac{1 \pm z_i}{2}.$$
2. (Final output) Output the resulting string $z' \in \{\pm 1\}^N \times \{\pm 1\}^N$.

Brief intuition. The main idea behind the choice of G' is that "Gaussian distributions are nice to work with". Thus, ideally we would like to analyze G' rather than this clunky "discrete" object D, which is necessary mainly due to the query model (e.g. recall a quantum query injects $z_i' \in \{\pm 1\}$ as a phase). Luckily, due to the choice of D, it turns out that as far as expectation values are concerned, both D and G' are "equivalent" in the following sense.

Exercise. Prove that $E[z_i'] = z_i$ for all $i \in [2N]$, where recall z_i' (z_i) are the discrete (continuous) coordinates.

3 Distinguishing D from U_{2N} is easy quantumly

The construction. Given oracle access to input $z' \in \{\pm 1\}^{2N}$, our goal is to decide whether z' was drawn according to D or U_{2N}. To do this quantumly, we imagine that the oracle for accessing z', denoted O_z, is split into two oracles O_x and O_y, such that for any $i \in [N]$,

$$(\text{"x part"}) \quad O_x|i\rangle \mapsto z_i'|i\rangle \qquad \text{and} \qquad (\text{"y part"}) \quad O_y|i\rangle \mapsto z_{N+i}'|i\rangle.$$

Exercise. Show how to implement O_x and O_y given O_z.

The quantum circuit for distinguishing D from U_{2N} is given in Figure 1. It is denoted V_n, since it acts on $n+1$ wires; the first wire is a control denoted c, and wires 2 to $n+1$ are fed to the oracle O_c. The control mechanism is as follows: If $c = 0$, then O_c applies O_x, and if $c = 1$, O_c applies O_y.

Correctness. For the remainder of this section, set $z' = x'y'$ for $x', y' \in \{\pm 1\}^N$. The following theorem shows that V_n works as intended.

Theorem 4. *Suppose circuit V_n of Figure 1 outputs 1 when it measures 1 and outputs -1 when it measures 0 (i.e. we are measuring observable $-Z$). Then, the expected output of V_n on $z' \sim U_{2N}$ (respectively, $z' \sim D$) is 0 (respectively, ϵ).*

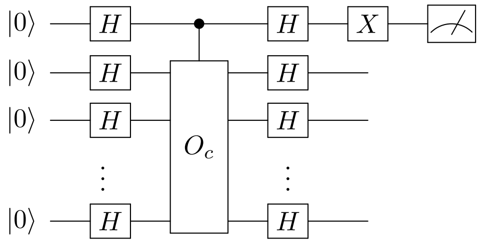

Figure 1: The quantum circuit V_n for distinguishing D from U_{2N}.

Exercise. Theorem 4 is in terms of expectation, but we defined "advantage" in Theorem 1 using probabilities. Why does the former immediately imply the quantum advantage claimed in Theorem 1? (Hint: No repetition of the protocol is needed. Use the fact that V_n has only two possible outcomes, ±1.)

Proof of Theorem 4. We proceed in three steps.

Step 1: The probability with which V_n outputs 1.

Exercise. Show that V_n measures 1 in the control qubit with probability

$$\frac{1}{2}\left(1+\left(\frac{1}{\sqrt{2^{3n}}}\sum_{i,j\in[N]}(-1)^{i\cdot j}x'_iy'_j\right)\right)=\frac{1}{2}\left(1+\left(\frac{1}{N}\sum_{i,j\in[N]}(H^{\otimes n}(i,j))x'_iy'_j\right)\right)=:\frac{1}{2}(1+\varphi(x',y')),$$

where $i\cdot j$ denotes the inner product modulo 2 of the bit strings i and j, and $H^{\otimes n}(i,j)$ is the (i,j)th entry of $H^{\otimes n}$. To characterize the acceptance probability of V_n, it hence suffices to analyze the multilinear polynomial $\varphi(x',y')$.

Exercise. Why is φ multilinear?

Step 2: Expected output of V_n for $z'\sim U_{2N}$.

Exercise. Prove that $E_{(x',y')\sim U_{2N}}[\varphi(x',y')]=0$. (Hint: There are three tools in mathematics which should always be at the top of your toolbox; the Cauchy-Schwarz inequality, Taylor series, and the linearity of expectation.)

We hence conclude that when $z'\sim U_{2N}$, the circuit V_n outputs each possible answer with probability 1/2.

Step 3. Expected output of V_n for $z'\sim D$. Recall in Section 2 that we had the gall to call discrete-valued distribution D "clunky", but the multivariate Gaussian distribution G' "nice". Here we will see why. First, we claim that

$$E_{(x',y')\sim G'}[\varphi(x',y')]=\epsilon, \tag{3}$$

where note we are using G'.

Exercise. Prove that for $(x',y')\sim G'$ and any $i,j\in[N]$, $E_{(x',y')\sim G'}[x'_iy'_j]=\epsilon\frac{(-1)^{i\cdot j}}{\sqrt{N}}$.

Via the exercise above and the linearity of expectation, the claim of Equation (3) follows:

$$E_{(x',y')\sim G'}[\varphi(x',y')]=\frac{1}{N}\sum_{i,j\in[N]}\frac{(-1)^{i\cdot j}}{\sqrt{N}}E_{(x',y')\sim G'}[x'_iy'_j]=\frac{1}{N^2}\sum_{i,j\in[N]}\epsilon=\epsilon.$$

So now we know φ has the right expectation with respect to G'; but we need the expectation of φ relative to D to correctly capture the expected output of V_n. For this, we use the following remarkable lemma (whose proof is omitted).

Lemma 5. *For any multilinear function $F : \mathbb{R}^{2N} \mapsto \mathbb{R}$,*

$$E_{z' \sim D}[F(z')] = E_{z \sim G'}[F(z)].$$

In words, if we process the input z' by a sufficiently restricted function F, namely multilinear F, then one cannot distinguish on expectation whether samples are drawn from D or G'.

Exercise. Combine Equation (3) and Lemma 5 to complete the proof of Theorem 4. What is the multilinear function F set to in our use of Lemma 5? □

4 Distinguishing D from U_{2N} is hard classically

We now wish to show that on expectation, any bounded depth circuit (as outlined in Theorem 1) cannot distinguish between samples z' drawn from D versus U_{2N}. A hint as to where we might wish to begin is given by Lemma 5, which recall says that on expectation under the action of any multilinear map F, D and U_{2N} are indistinguishable. Lucky for us, it is well-known that the action of any Boolean circuit is captured by a multilinear map, which we now review.

4.1 Boolean circuits and multilinear polynomials

Let C be an arbitrary circuit mapping $\{\pm 1\}^n$ to $\{\pm 1\}$. We can equivalently view C as a Boolean function from $\{\pm 1\}^n$ to $\{\pm 1\}$, to which the following lemma applies.

Lemma 6. *Let $f : \{\pm 1\}^n \mapsto \{\pm 1\}$ be a Boolean function. Then, f has a (unique) multilinear extension $g : \mathbb{R}^n \mapsto \mathbb{R}$ such that $g(x) = f(x)$ when $x \in \{\pm 1\}^n$, and*

$$g(x) = \sum_{S \subseteq [n]} \widehat{g}(S) \cdot \prod_{i \in S} x_i,$$

for $\widehat{g}(S) \in \mathbb{R}$ the Fourier coefficients *of g.*

Proof sketch. The idea is to write g as a sum of 2^n terms, each of which encodes the product of an output $f(x)$ for some x times an "indicator function" which eliminates all terms other than x. The construction is best sketched via an example. Let $n = 3$, and consider input $x = (1, -1, 1)$. Then, we add to g the term

$$\left(\frac{1 + x_1}{2}\right)\left(\frac{1 - x_2}{2}\right)\left(\frac{1 + x_3}{2}\right) f(x).$$

The three terms in brackets serve as the "indicator function" alluded to above, which is uniquely specified by the string $x = (1, -1, 1)$.

Exercise. Verify that when $x = (1, -1, 1)$ is plugged into the equation above, the output is $f(x)$. What is the output when $x \in \{\pm\}^3$ but $x \neq (1, -1, 1)$?

Exercise. What would be the indicator function for, say, $x = (-1, -1, -1)$?

By adding to g a term of the above form for each possible input $x \in \{\pm 1\}^n$, expanding the brackets and simplifying the resulting expression, we obtain the final desired polynomial g, which may consist of exponentially many terms in general.

Exercise. Why is g multilinear? □

4.2 Tools and proof approach

In the remainder of our discussion, let us henceforth use C interchangeably to mean a classical circuit and its multilinear extension (Lemma 6), where the desired interpretation will hopefully be clear from context. Our goal is to show that if $C : \{\pm 1\}^{2N} \mapsto \{\pm 1\}$ is constant depth and has size $2^{\log^c N}$ for constant c, then C cannot distinguish on expectation between D and U_{2N}, i.e.

$$|E_{z' \sim D}[C(z')] - E_{u \sim U_{2N}}[C(u)]| \leq \frac{\text{polylog}(N)}{\sqrt{N}},$$

where recall $N = 2^n$ is exponentially large in the input parameter, n.

Basic proof approach. We begin with a simple observation.

Exercise. Prove that $E_{u \sim U_{2N}}[C(u)] = \widehat{C}(\emptyset)$, for $\widehat{C}(\emptyset)$ the Fourier coefficient of C corresponding to empty set $S = \emptyset$.

In words, the expectation of the circuit C under U_{2N} simply eliminates all higher order Fourier coefficients, leaving behind the constant term in the Fourier expansion of C, $\widehat{C}(\emptyset)$. Thus, it suffices for our goal to show

$$\left| E_{z' \sim D}[C(z')] - \widehat{C}(\emptyset) \right| \leq \text{“small”}. \tag{4}$$

Our basic proof approach is hence to show that, on expectation, the contribution of higher order Fourier coefficients to $C(z')$ on inputs $z' \sim D$ is bounded.

Tools. To follow this basic approach, we require a pair of tools.

1. *Tail bounds on Fourier coefficients (proof omitted):*

   Lemma 7. *Let $C : \{\pm 1\}^{2N} \mapsto \{\pm 1\}$ be a Boolean circuit of depth d and of size s. Then for any $k \in \mathbb{N}$, there exists $c \in O(1)$ such that*
   $$\sum_{S \subseteq [2N]} \left| \widehat{C}(S) \right| \leq (c \log s)^{(d-1)k}.$$

   In words, Lemma 7 intuitively says that for constant depth circuits, the low order (e.g. $k \in O(1)$) Fourier coefficients are bounded. Thus, this lemma alone gets us part of the way to Equation (4); the problem is the lemma does *not* work *a priori* for higher order coefficients.

2. *Random walks:* To use Lemma 7 to also bound the high-order Fourier coefficients of C, we use a trick — it turns out we can simulate drawing $z' \sim D$ with a random walk which takes “baby steps”. Applying Lemma 7 to each such “baby step” and applying the triangle inequality then does the trick.

4.3 Main theorem and proof sketch

We are now ready to state and sketch a proof of the main theorem, which is a quantitative version of Equation (4).

Theorem 8 (D fools bounded depth circuits)**.** *Let $C : \{\pm 1\}^{2N} \mapsto \{\pm 1\}$ be a Boolean circuit of depth d and size s. Then*

$$\left| E_{z' \sim D}[C(z')] - \widehat{C}(\emptyset) \right| \leq \frac{12\epsilon (c \log s)^{2(d-1)}}{\sqrt{N}}.$$

To prove Theorem 8, we require one final lemma, which is proven using the tail bounds of Lemma 7 (i.e. we will not directly use Lemma 7 otherwise in this lecture).

Lemma 9. *Fix $p \leq 1/4$. Let $C : \{\pm 1\}^{2N} \mapsto \{\pm 1\}$ be a Boolean circuit of depth d and size s such that $\sqrt{\epsilon} p (c \log s)^{d-1} \leq \frac{1}{4}$. Then, for any $z_0 \in [-\frac{1}{2}, \frac{1}{2}]$,*

$$|E_{z \sim G'}[C(z_0 + pz)] - C(z_0)| \leq \frac{12\epsilon p^2 (c \log s)^{2(d-1)}}{\sqrt{N}}.$$

Intuitively, Lemma 9 says that if in a random walk we evaluate C at our "current point" z_0, versus if evaluate C on a slight perturbation of order p of z_0 (i.e. on $z_0 + pz$), then the ability for C to distinguish these inputs is damped quadratically in p. Thus, to simulate drawing $z' \sim D$, we shall run a random walk with baby steps of small order, i.e. $p = 1/\sqrt{N}$.

Proof sketch of Theorem 8. Assume without loss of generality that $\sqrt{\epsilon}(c \log s)^{d-1} \leq \frac{1}{4} N^{1/4}$, as otherwise the claim is vacuous. We run the following random walk for $t = N$ steps, with perturbation size $p = \frac{1}{\sqrt{N}}$:

1. Draw t samples $z^{(1)}, \ldots, z^{(t)} \sim G'$.
2. Set the output of "step i" of the random walk, for $i \in \{0, \ldots, t\}$, to be random variable

$$z^{\leq (i)} = p(z^{(1)} + \cdots + z^{(i)}).$$

One can show that $z^{\leq (t)} \sim G'$, i.e. the random walk exactly reproduces the distribution G'. (Namely, $z^{\leq (i)}$ and G' share the same expectation and covariance matrix.)

Now, for any step $i \in \{0, \ldots, t-1\}$, since each entry of $z^{\leq (i)}$ is Gaussian with variance scaling as $\sim p^2 \epsilon$, we have with high probability that $z^{\leq (i)} \in [-1/2, 1/2]^{2N}$, as required for Lemma 9. Thus, by setting $z_0 = z^{\leq (i)}$ and calling Lemma 9 (i.e. this is where we use the tail bounds), we have that

$$\left| E[C(z^{\leq (i+1)})] - E[C(z^{\leq (i)})] \right| \leq \frac{12\epsilon p^2 (c \log s)^{2(d-1)}}{\sqrt{N}}. \tag{5}$$

Thus, the change in output of C, on expectation from step i to $i+1$ of the walk, is bounded by a factor scaling with $p^2 = 1/N$. Applying this recursively over all steps of the walk, we conclude:

$$\begin{aligned}
\left| E_{z' \sim D}[C(z')] - \widehat{C}(\emptyset) \right| &= \left| E_{z \sim G'}[C(z)] - \widehat{C}(\emptyset) \right| \\
&= \left| E[C(z^{\leq (t)})] - \widehat{C}(\emptyset) \right| \\
&\leq \sum_{i=0}^{t-1} \left| E[C(z^{\leq (i+1)})] - E[C(z^{\leq (i)})] \right| \\
&\leq t \frac{12\epsilon p^2 (c \log s)^{2(d-1)}}{\sqrt{N}} \\
&= \frac{12\epsilon (c \log s)^{2(d-1)}}{\sqrt{N}},
\end{aligned}$$

where the first statement follows from Lemma 5, the second since $z^{\leq (t)} \sim G'$, the third by the triangle inequality over all steps of the walk, and the fifth by Equation (5).

Exercise. The third statement above involves a term of form $\left| E[C(z^{\leq (1)})] - E[C(z^{\leq (0)})] \right|$, but we have not explicitly defined $z^{\leq (0)}$. Intuitively, we require $E[C(z^{\leq (0)})] = \widehat{C}(\emptyset)$. How should we define $z^{\leq (0)}$ for this requirement to hold? (Hint: Which input $x \in \mathbb{R}^{2N}$ reduces the multilinear extension C to its constant term, $\widehat{C}(\emptyset)$?) Why does this choice of $z^{\leq (0)}$ make sense given our definitions of $z^{\leq (i)}$ for $i \in [t]$? □