CHAPTER 2

Improved separation between quantum and classical computers for sampling and functional tasks

Quantum computers are thought to be poised to outperform classical computers on a number of significant tasks, such as integer factorisation or simulation of quantum systems. Despite this widely held belief, we have no formal proof that factoring is not solvable in polynomial time on a classical computer, or even that $\mathsf{BQP} \neq \mathsf{BPP}$.

This is perhaps not surprising given the challenge posed by proving unconditional results in complexity theory; famously $\mathsf{P} \overset{?}{=} \mathsf{NP}$ remains unresolved after decades of study. A more realistic approach is to base the hardness of quantum computing on widely accepted complexity-theoretic conjectures.

This approach has so far been quite successful for sampling problems, with a number of papers [17, 18] showing that various quantum sampling experiments would be hard for a classical algorithm unless the polynomial hierarchy collapses to its third level, or else a widely believed conjecture fails. These results are especially interesting as they naturally imply that quantum computers could implement functions classical computers could not ($\mathsf{FBQP} \neq \mathsf{FBPP}$) due to a surprising equivalence between sampling and functional classes [14].

Similarly, there has been a body of work showing that various changes to quantum computing make quantum computing vastly more powerful than similarly modified classical computing. A well-known example is $\mathsf{PostBQP}$ vs $\mathsf{PostBPP}$, i.e. quantum/classical computing with postselection. Surprisingly, the former is in some sense equivalent to problems involving exact counting (PP) [19], while the latter is only equivalent to approximate counting [20]. This disconnect implies that the classes are likely not equal, indeed if they are this would also imply a collapse of the polynomial hierarchy to the third level [19].

Both of these research lines provide strong evidence for that quantum computing is more powerful than classical computing, but they rely on a number of conjectures and complexity theory conditions. We are therefore motivated to attempt to minimise or remove the need for these conjectures/conditions until we are left with compelling evidence that quantum computing is fundamentally stronger than classical computing.

Hope that weaker conditions might be possible was provided by Fujii et al. [21], who showed that if classical computers can *exactly sample* from any one of a number of quantum advantage experiments then the polynomial hierarchy would collapse to its second level, an improvement over the existing collapse to third level. In fact they push the collapse all the way to AM, in the second level of the hierarchy. While this result provides insight as to what may be possible, it stops short of our goal as the proof strategy did not extend to additive approximate sampling, which is arguably the realistic case [17] and is relevant for questions such as $\mathsf{FBQP} \overset{?}{=} \mathsf{FBPP}$.

In this chapter, we provide the following new complexity theoretic result, which we then use to improve the abovementioned results to a 2nd level polynomial hierarchy collapse. In the following theorem, we use $\|$ to denote only one round of parallel oracle calls (i.e. many oracle calls can be asked, but must all be asked in a single round, without adapting to the previous oracle calls).

Theorem (Main)
If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}_{\|}$ *then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$

By showing that the existing collapse results for classical boson sampling, commuting circuit sampling and 1 clean qubit sampling can all be modified to show $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}_{\|}$ (we are required to add the parallel requirement to $\mathsf{BPP}^{\mathsf{NP}}_{\|}$ for our collapse), we improve the implied collapse of the polynomial hierarchy from 3rd order to 2nd. Consequently, this strengthens the evidence that $\mathsf{FBQP} \neq \mathsf{FBPP}$. This theorem can also be made to strengthen the hierarchy collapse implied by assuming $\mathsf{PostBQP} = \mathsf{PostBPP}$ from the third to the second level.

From a purely complexity-theoretic perspective, there is an interesting alternative form of this result in terms of exact and approximate counting[1].

$$\text{If } \mathsf{P}_{\|}^{\mathsf{ExactCount}} = \mathsf{P}_{\|}^{\mathsf{ApproxCount}} \text{ then } \mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$$

This naturally gives our main result the following interpretation: *If the problems solved with the aid of exact counting can also be solved with approximate counting, then both are contained in* $\mathsf{ZPP}^{\mathsf{NP}}$. The generality of this interpretation hints that there may be applications of this theorem to counting beyond the quantum theory considered herein. It is also interesting to note that the proof contained herein does not relativise as we rely on the checkability of $\#\mathsf{P}$ for a step of our proof.

The chapter is structured in 3 sections. Section 2.1 focuses on the proof of our central theorem, section 2.2 proves that the polynomial hierarchy collapses to second order if $\mathsf{PostBQP} = \mathsf{PostBPP}$. In the third section, we prove our various sampling results. Finally, we close by discussing interesting future research and open questions.

2.1 Main theorem

In this section we prove our main theorem:

Theorem 2.1.1
If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ *then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$.

We will assume readers are broadly familiar with classes such as $\#\mathsf{P}$, where classes are undefined we use the definitions given in the complexity zoo [22]. We use the notation $\mathsf{A}_{\|}^{\mathsf{B}}$ for A with one set of parallel oracle calls to B.

It is useful to first give an informal description of the proof of theorem 2.1.1 before we proceed to the formal proof to give the reader a better idea of the purpose of each lemma. Broadly the proof relies on the notion of random-reducibility, where a given instance of a problem can be solved by a polynomial time algorithm with randomly selected instances of some other problem. If a problem can be randomly reduced to itself we say it is random self-reducible. The first key realisation for Theorem 2.1.1 is that a random self-reducible language that is in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ is itself random reducible to NP, because randomly selecting x for a language in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ gives rise

[1]Exact counting can be defined as exactly counting the number of accepting inputs of some poly-time machine, approximate counting is getting within a multiplicative factor of 2 of exact counting (equivilently to within a $1 + \frac{1}{poly(n)}$ factor).

to random set of non-adaptive calls oracle calls to NP, which meets the definition of random reducibility to NP. Next, as #P is random self-reducible [23], the antecedent of Theorem 2.1.1 implies that it is random reducible to NP. The second key realisation is to notice that the proof by Feigenbaum and Fortnow that NP cannot be random self-reducible unless it is also in coNP/poly extends beyond self reducibility; any language that is random reducible to NP is in coNP/poly. By repeating the argument for coNP we show $\#\mathsf{P} \subseteq \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. Arvind and Köbler [24] showed any checkable language in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ is low for Σ_2^{P} which gives us a polynomial hierarchy collapse to Σ_2^{P}. The final collapse to $\mathsf{ZPP}^{\mathsf{NP}}$ comes from using the $\mathsf{BPP}_{||}^{\mathsf{NP}}$ algorithm to produce proofs verifiable in P^{NP}, thereby achieving zero error.

We begin by defining random reducibility. A problem is randomly self-reducible if any given instance of the problem can be solved probabilistically by a polynomial time algorithm with access to solved random instances of the same problem. For our purposes we will work with a similar concept: random-reducibility, which is the same except the random instances may be from a different problem. It should be stated that this is different from a 'randomized reduction', which is a randomised Karp reduction. Random reductions are sometimes called 1-locally random reductions to avoid this confusion.

Definition 2.1.1 (Random Reducible)
A function f is ***nonadaptively*** *$k(n)$* ***random-reducible*** *to a function g if there are polynomial time computable functions σ and ϕ with the following two properties.*

(1) For all n and all $x \in \{0,1\}^n$ $f(x)$ can be solved by ϕ with instance of $g(y)$ for y selected by σ with high probability:

$$\Pr_r \left(f(x) = \phi(x, r, g(\sigma(1,x,r)), \dots, g(\sigma(k,x,r))) \right) \geq 2/3$$

(2) For all n, all pairs $\{x_1, x_2\} \subset \{0,1\}^n$ and all i, if r is chosen uniformly at random then $\sigma(i, x_1, r)$ and $\sigma(i, x_2, r)$ are identically distrubuted.

If both conditions hold we say (σ, ϕ) is a random-reduction from f to g.

As we are dealing with no other forms of random reducibility we will just say 'f is rr to g' when f is non-adaptively $k(n)$ random-reducible to g for some polynomial k. If a function is rr to itself then we say the function is random self-reducible (rsr). A set is rr to another if its characteristic function is rr. A set of languages, A is rr to B if every $L \in \mathsf{A}$ is rr to some $L' \in \mathsf{B}$.

As with probabilistic classes like BPP, we can boost random reducibility an arbitrarily low (2^{-n}) failure probability.

Lemma 2.1.1 ([23])
If function f is non-adaptively $k(n)$ random-reducible to g then for any polynomial, $t(n)$, f is non-adaptively $24t(n)k(n)$ random-reducible to g where condition (1) holds for at least $1-2^{-t(n)}$ of the r's in $\{0,1\}^{24t(n)k(n)}$ (probability $1-2^{-t(n)}$ of success).

Remark
In general, this boosting may not work for a definition of random reducibility dealing with relations instead of functions as they may have multiple correct outputs. However, counting problems like Perm, *which has only one correct output for a given input, can be boosted.*

Lemma 2.1.2 ([23])
Any $\#\mathsf{P}$-complete language is rsr.

This concludes our definitions, we can now state an intermediate theorem to our final result.

Theorem 2.1.2
If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then $\mathsf{P}^{\#\mathsf{P}}$ is random reducible to NP.

Before we prove this, we will provide a number of smaller results. In the following lemma the notation $\mathsf{A}^{(\mathsf{B}[1])}$ means A with one oracle call to B.

Lemma 2.1.3
Any language in $\mathsf{P}^{\#\mathsf{P}[1]}$ is random reducible to $\#\mathsf{P}$.

Proof. As $\#\mathsf{P}$-complete languages are random self-reducible and only one oracle call is needed we can use a random reduction of $\#\mathsf{P}$ to give the answer to the oracle call and use the polynomial time ϕ algorithm to perform the rest of the P algorithm. $\square$

The next lemma captures the intuition that $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ algorithms are 'almost' rr to NP, only missing the random element selection part of the definition (property (2)).

Lemma 2.1.4
For all L in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ there exists polynomial time computable functions σ_B, ϕ_B such that

$$\Pr_r\left(L(x) = \phi_B(x, r, \mathsf{SAT}(\sigma_B(1,x,r)), \ldots, \mathsf{SAT}(\sigma_B(m,x,r)))\right) \geq 1 - 2^{-n}.$$

Proof. If $L \in \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then there exists a polynomial time algorithm, A, to decide $x \in L$ which calls only one set of up to m non-adaptive NP queries. We assume the NP calls SAT for simplicity. Let the algorithm for $\sigma_B(i, x, r)$ run A until the step involving oracle queries and then output just the i'th query (assuming some arbitrary query ordering). The algorithm for ϕ_B is now just the algorithm A using the oracle calls produced asked by $\sigma_B(i, x, r)$ in place of directly making its own calls. Since both ϕ_B and σ_{B} have access to the same random string r they will both select the same oracle calls. □

Lemma 2.1.4 is useful as it proves that $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ fulfils property (1) of the definition of random reducibility to NP, but does not prove the random distribution of $\sigma_B(1, x, r)$. The next theorem shows that if a langauge already randomly reduces to $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ then this random reduction condition (condition (2)) is also fufilled, and thus the language is random reducible to NP

Lemma 2.1.5
All languages random reducible to a language in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ *are random reducible to* NP*.*

Proof. Let L be a language which is random reducible to $L_B \in \mathsf{BPP}_{\|}^{\mathsf{NP}}$. We will demonstrate that L is rr to NP by writing out the definition of random reducibility to L_B then using lemma 2.1.4 to substitute the calls to L_B with a formula using calls to SAT. We can then rearange this into a format which directly makes calls to SAT and we show that these calls inherit the randomness property from the random reduction of L to L_B.

Let us assume the random reduction of L to L_B on input x uses k calls[2], from the definition of random reducibility there exists ϕ and σ such that

$$\Pr_r\left(L(x) = \phi(x, r, L_{\mathsf{B}}(\sigma(1, x, r)), \ldots, L_B(\sigma(k, x, r)))\right) \geq 1 - 2^{-n}.$$

Define $y_i := \sigma(i, x, r)$.

$$\Pr_r\left(L(x) = \phi(x, r, L_{\mathsf{B}}(y_1), \ldots, L_{\mathsf{B}}(y_k))\right) \geq 1 - 2^{-n}$$

Using lemma 2.1.4 we can substitute L_B with ϕ_B, σ_B and a random string r_i for the i'th call to L_B. In the following we assume each lemma-2.1.4-reduction uses m SAT calls.

[2]an input of length n can only use some polynomial number of oracle calls, upper bounded by k

$$\Pr_{r,r_1,\ldots r_k}\big(L(x) = \phi(x,r,\phi_B(y_1,r_1,\mathsf{SAT}(\sigma_B(1,y_1,r_1)),\ldots,\mathsf{SAT}(\sigma_B(m,y_1,r_1))),$$
$$\ldots,\phi_B(y_k,r_k,\mathsf{SAT}(\sigma_B(1,y_k,r_k)),\ldots,\mathsf{SAT}(\sigma_B(m,y_k,r_k))))\big)$$
$$\geq 1-(k+1)2^{-n}.$$

The failure probability $1-(k+1)2^{-n}$ comes the failure probability of the k reductions combined with the failure probability from the random reduction.

We can now begin to combine elements of the reduction from L to L_B with reduction from L_B to SAT. We start with ϕ which we combine with ϕ_B to define ϕ', informally we can think of this as doing the two polynomial-time algorithms as a larger polynomial time algorithm.

$$\phi'(x,r,r_1,\ldots,r_k,\mathsf{SAT}(\sigma_\mathsf{B}(1,\mathsf{y}_1,\mathsf{r}_1)),\ldots\mathsf{SAT}(\sigma_\mathsf{B}(\mathsf{m},\mathsf{y}_\mathsf{k},\mathsf{r}_\mathsf{k}))) :=$$
$$\phi(x,r,\phi_B(y_1,r_1,\mathsf{SAT}(\sigma_B(1,y_1,r_1)),\ldots,\mathsf{SAT}(\sigma_B(m,y_1,r_1))),\ldots,$$
$$\phi_B(y_k,r_k,\mathsf{SAT}(\sigma_B(1,y_k,r_k)),\ldots,\mathsf{SAT}(\sigma_B(m,y_k,r_k))$$

We use '$x\#y$' to denote y appended to x. Define $\mathbf{r} := r\#r_1\#\ldots\#r_k$.

Again we define a new function, σ', as the combination of two existing function, σ and σ_B:

$$\sigma'(i,j,x,\mathbf{r}) := \sigma_B(i,\sigma(j,x,r),r_j).$$

Which gives the following simplified equation:

$$\Pr_{\mathbf{r}}\big(L(x) = \phi'(x,\mathbf{r},\mathsf{SAT}(\sigma'(1,1,x,\mathbf{r})),\ldots\mathsf{SAT}(\sigma'(m,k,x,\mathbf{r}))\big) \tag{2.1}$$
$$\geq 1-(k+1)2^{-n}. \tag{2.2}$$

We will now directly check the definition of σ', ϕ' and the equation 2.1 against the definition of random-reducbile, recall the two conditions definition 2.1.1:

(1) For all n and all $x \in \{0,1\}^n$ $f(x)$ can probably be solved by ϕ with instance of $g(y)$ for y selected by σ:

$$\Pr_r\big(f(x) = \phi(x,r,g(\sigma(1,x,r)),\ldots,g(\sigma(k,x,r)))\big) \geq 2/3$$

(2) For all n, all pairs $\{x_1,x_2\} \subset \{0,1\}^n$ and all i, if r is chosen uniformly at random then $\sigma(i,x_1,r)$ and $\sigma(i,x_2,r)$ are identically distrubuted.

We can see that these two conditions are met:

(1) For sufficiently large n, $1-(k+1)2^{-n} > 2/3$ therefore equation 2.1 is of the form:

$$\Pr_r\left(f(x) = \phi(x, r, \mathsf{SAT}(\sigma(1,x,r)), \ldots, \mathsf{SAT}(\sigma(k,x,r)))\right) \geq 2/3.$$

(2) In short: a composition of i.d. maps is an i.d. map. For all n, all pairs $\{x_1, x_2\} \subset \{0,1\}^n$, all i and j, if r is chosen uniformly at random then $\sigma(j, x_1, r)$ and $\sigma(j, x_2, r)$ are identically distributed (as per their definition), therefore $\sigma'(i\#j, x_{1/2}, \mathbf{r})$ is identically distributed too. This is because the only dependence on x is through the identically distributed σ: $\sigma'(i\#j, x, \mathbf{r}) = \sigma_{\mathsf{B}}(i, \sigma(j,x,r), r_j)$. This holds for all r_i, r_j.

Thus fufilling the conditions for L to be random reducible to NP □

Finally, we can proceed with the proof of theorem 2.1.2

Proof of theorem 2.1.2. Toda [25] showed that $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}[1]}$. Therefore, by the assumption of the theorem if $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}} = \mathsf{PH}$ then $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\#\mathsf{P}[1]}$.

By lemma 2.1.3 we know $\mathsf{P}^{\#\mathsf{P}}$ is rr to $\#\mathsf{P}$. Under the assumption of this theorem, $\#\mathsf{P} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$, $\mathsf{P}^{\#\mathsf{P}}$ is rr to $\mathsf{BPP}_{\|}^{\mathsf{NP}}$. By lemma 2.1.5 this means $\mathsf{P}^{\#\mathsf{P}}$ is rr to NP. □

The next result is heavily based on the work of Feigenbaum and Fortnow [23]. They show that if NP is random self reducible then coNP is in $\mathsf{NP/poly}$, while the result we are trying to show is slightly different than this, the method is very similar.

Theorem 2.1.3
Any language that is random-reducible to a language in NP *is also in* $\mathsf{coNP/poly} \cap \mathsf{NP/poly}$.

Proof. Let $\mathsf{AM}^{\mathsf{poly}}$ be AM with polynomial advice given to Arthur[3]. It turns out that $\mathsf{AM}^{\mathsf{poly}} = \mathsf{NP/poly}$ and $\mathsf{coAM}^{\mathsf{poly}} = \mathsf{coNP/poly}$ [23]. We will prove L is in $\mathsf{AM}^{\mathsf{poly}} \cap \mathsf{coAM}^{\mathsf{poly}}$, beginning with $L \in \mathsf{AM}^{\mathsf{poly}}$.

Suppose $L(x)$ is non-adaptively $k(n)$ random-reducible to $\mathsf{SAT}(x)$ with error probability 2^{-n}. We will adopt the notation $y_i = \sigma(i, x, r)$. For

[3]This is not the same as $\mathsf{AM/poly}$, as the latter must be an AM language for all advice, whereas the former only needs to be defined for the correct advice.

instance size n we will give Arthur the advice $(p_1, \ldots, p_k)$ where p_i is the probability that $y_i = 1$,

$$p_i = \Pr_r(\mathsf{SAT}(\sigma(i, x, r)) = 1).$$

As $\sigma(i, x, r)$ is distributed identically (given uniform random r) for all x this advice does not depend on the input x.

The proof system will consist of Arthur selecting $m = 9k^3$ random strings, $\{r_j\}_{j\in[m]}$, and passing this to Merlin, these random strings defines which SAT queries, $y_{0,j}, \ldots, y_{k,j}$, will be made. Merlin will hand back m 'transcripts'. These transcripts consist of a list $w_{i,j}$ which is either a witness for $y_{i,j}$ or is the string 'NIL' (which is interpreted as the claim that $y_{i,j} \notin \mathsf{SAT}$). For each of the m random strings we get the transcript:

$$Transcript(x, r_j) = (w_{0,j}, w_{1,j}, \ldots, w_{k,j})$$

Arthur performs three checks:

(1) For all i, j either the witness $w_{i,j}$ is 'NIL' or he checks the witness is valid for $y_{i,j} \in \mathsf{SAT}$

(2) Let $b_{i,j}$ be 0 if $w_{i,j}$ is 'NIL' and one otherwise. For all $j \in [m]$ Arthur checks that $\phi(x, r_j, b_{0,j}, \ldots, b_{k,j}) = 1$, i.e. If Merlin has told the truth then ϕ will accept.

(3) For each $i \in [k]$ Arthur checks that more than $p_i m - 2\sqrt{km}$ of the $y_{i,j}$ have been proved to be in SAT, if this condition does not hold he rejects.

If these checks all pass, then Arthur accepts. The trick in this proof is this final condition. Over the random strings, each y_i has some probability of being in SAT with a given variance, by picking m to be large we force this variance to be small. With high probability, a correct answer lies in this range. Since Merlin cannot lie about yes instances (since he must prove them), he can only lie about no instances. However, if he lied too much it would be clear that not enough of y_i are in SAT, thus we could probabilistically guess he was cheating. To exploit this, m is picked precisely so Merlin cannot cheat on all j without exceeding his 'lying budget'. We will now formally show soundness and completeness.

Completness. If $x \in L$ and Merlin is honest, providing witnesses for all $y_{i,j}$ in SAT, condition (1) will always hold. Condition (2) holds with probability at least $1 - 2^{-n}$ (by the definition of rr). Therefore we just need to show condition (3) holds with high probability.

Let $Z_{i,j} := (y_{i,j} = 1)$ and $Z_i = \sum_{j=1}^{m} Z_{i,j}$. We defined our advice so $p_i = \mathbb{E}[Z_i]/m$, we can derive that $\mathrm{Var}(X) = p_i(1 - p_i)m < m$. We can use these properties and Chebyshev's inequality to bound our probability of failing check (2).

$$\begin{aligned} &\Pr\left(Z_i \leq p_i m - 2\sqrt{km}\right) \\ \leq &\Pr\left(\|Z_i - p_i m\| \geq 2\sqrt{km}\right) \\ = &\Pr\left(\|Z_i - \bar{Z}_i\| \geq 2\sqrt{km}\right) \\ \leq &\mathrm{Var}(X)/4km \leq 1/(4k) \leq 1/4 \end{aligned}$$

The probability of failing any check given $x \in L$ is less than $\frac{1}{4} + 2^{-n}$ which is less than $1/3$ for large enough n. This proves completeness

Soundness. Suppose $x \notin L$, if Arthur accepts then all 3 checks must have passed for all $j \in [m]$. If some $y_{i,j}$ is in SAT but Merlin has provided $w_{i,j} =$ 'NIL' then we say Merlin has lied about $y_{i,j}$, if check (1) passes Merlin has not lied about any 'yes' instances, so we disregard this case. The probability of the random reduction failing is 2^{-n} without any lies, so for all m reductions to fail Merlin must lie m times with probability $(1 - 2^{-n})^m > 1 - m2^{-n}$.

If Merlin lies m times there must be an i that he claims at least m/k of the $y_{i,j}$ are not in SAT when they are. To satisfy condition (3) at least $p_i m - 2\sqrt{km}+\mathbf{m/k}$ of the $y_{i,j}$ must be in SAT to leave 'room' for Merlin to lie m/k times without violating (3). The probability of this is given by a Chernoff bound on the random variable Z_i, as defined above.

$$\begin{aligned} &\Pr\left(Z_i \geq p_i m - 2\sqrt{km} + m/k\right) \\ = &\Pr\left(Z_i - p_i m \geq m/k - 2\sqrt{km}\right) \\ = &\Pr\left(Z_i - p_i m \geq 9k^3/k - 2\sqrt{k9k^3}\right) \\ = &\Pr\left(Z_i - p_i m \geq 3k^2\right) \\ \leq &e^{-2\times 9k^4/9k^3} = e^{-2\times k} \\ \leq &1/(4k) \end{aligned}$$

Combining this with the normal probability that the random reduction fails even with correct oracle answers (which had a 2^{-n} probability) gives an acceptance probability given $x \notin L$ of less than $m2^{-n} + 1/(4k) < 1/3$

for large enough n. This proves soundness.

The previous analysis can just as easily be repeated for $\mathsf{coAM}^{\mathsf{poly}}$ with Merlin proving no instances, giving $L \in \mathsf{AM}^{\mathsf{poly}} \cap \mathsf{coAM}^{\mathsf{poly}}$. By Feigenbaum and Fortnow we know this equals $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. □

To prove the next Theorem we must recall a crucial result by Arvind and Köbler: checkability [22, 24]. The notion of checkability is somewhat involved but intuitively has to do with whether efficent programs that decide the set can themselves be efficently checked. Since we just need the fact that PP and $\#\mathsf{P}$ are checkable to deploy the following theorem we will not formally define checkability, instead we refer the reader to [24, 26].

Theorem ([24], Theorem 22)
Checkable sets in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ *are low for* Σ_2^{P}.

The last result may seem unnecessary as it is well known that all sets which are in $\mathsf{NP} \subset (\mathsf{NP} \cap \mathsf{coNP})/\mathsf{poly}$ are also in $\mathsf{ZPP}^{\mathsf{NP}}$ [27], however the proof of this does not seem to carry over to $\mathsf{NP} \subset (\mathsf{NP/poly}) \cap (\mathsf{coNP/poly})$, and only with checkability can we get our set in $\mathsf{ZPP}^{\mathsf{NP}}$

Lemma 2.1.6
If $\mathsf{P}^{\#\mathsf{P}} \subset (\mathsf{coNP/poly} \cap \mathsf{NP/poly})$ *then* $\mathsf{P}^{\#\mathsf{P}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$

Proof. By Toda $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}}$ [25]. If P^{PP} is in $\mathsf{coNP/poly} \cap \mathsf{NP/poly}$ then clearly so is PP, combining this with the existence of PP-complete checkable sets [24] we know that PP is low for Σ_2^{P}.

This lowness implies we can put $\mathsf{P}^{\#\mathsf{P}}$ in Σ_2^{P} by the following series of inclusions

$$\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}} \subseteq (\Sigma_2^{\mathsf{P}})^{\mathsf{PP}} = \Sigma_2^{\mathsf{P}}.$$

As $\Sigma_2^{\mathsf{P}} \subseteq \mathsf{P}^{\#\mathsf{P}}$ by Toda [25] we can fix the previous inclusion into an equality: $\Sigma_2^{\mathsf{P}} = \mathsf{P}^{\#\mathsf{P}}$.

As $\Pi_2^{\mathsf{P}} \subseteq \mathsf{P}^{\#\mathsf{P}}$ (Toda [25]) we get $\Pi_2^{\mathsf{P}} \subseteq \Sigma_2^{\mathsf{P}}$. Which gives us the final equality:

$$\mathsf{P}^{\#\mathsf{P}} \subseteq \Sigma_2^{\mathsf{P}} \subseteq \mathsf{P}^{\Pi_2^{\mathsf{P}}} = \mathsf{P}^{\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$$

□

The previous lemma has achieved a collapse to the second level but we can collapse to $\mathsf{ZPP}^{\mathsf{NP}}$ by using the proveability of languages in $\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ to make the $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ algorithm zero-error.

Lemma 2.1.7
All languages in $\mathsf{BPP}^{\mathsf{NP}} \cap \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ *are in* $\mathsf{ZPP}^{\mathsf{NP}}$

Proof. Fix some $L \in \mathsf{BPP}^{\mathsf{NP}} \cap \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$. We can check if $x \in L$ using the $\mathsf{BPP}^{\mathsf{NP}}$ algorithm. Use the $\mathsf{BPP}^{\mathsf{NP}}$ program to determine a proof of this fact for either the Σ_2^{P} verifier or the Π_2^{P} verifier. We can test either proof in P^{NP} using the verifiers from either Σ_2^{P} or Π_2^{P}. With probability $1 - 2^{-n}$ the $\mathsf{BPP}^{\mathsf{NP}}$ algorithm succeeds in producing a valid proof in polynomial time, any valid proof proves or disproves $x \in L$. Therefore in polynomial time the algorithm has successfully determined if $x \in L$ with probability $1 - 2^{-n}$ and never incorrectly answers, placing $L \in \mathsf{ZPP}^{\mathsf{NP}}$. □

This completes the proof of our main theorem which is given below.

Theorem
If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ *then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$

Summary of proof. If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then $\mathsf{P}^{\#\mathsf{P}}$ is random reducible to NP (Theorem 2.1.2).

If $\mathsf{P}^{\#\mathsf{P}}$ is random reducible to NP then $\mathsf{P}^{\#\mathsf{P}}$ is in $\mathsf{coNP/poly} \cap \mathsf{NP/poly}$ (Theorem 2.1.3).

If $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{coNP/poly} \cap \mathsf{NP/poly}$ then $\mathsf{P}^{\#\mathsf{P}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ (Lemma 2.1.6).

If $\mathsf{P}^{\#\mathsf{P}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ and $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$ then $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$ (Lemma 2.1.7). □

Remark
Interestingly we can perform all of the oracle calls required to produce the proof of $x \in L$ in one parallel step, it then only takes one extra Oracle call to check this proof. Thus it is possible to answer correctly with probability $1 - 2^{-n}$ or with 'I don't know' after only 2 rounds of parallel NP *oracle calls.*

Remark
Feigenbaum and Fortnow [23] showed their proof applies beyond non-adaptive random self-reducibility, up to logarithmically many adaptive steps are allowed. For the same reasons, our proof could be extended to logarithmically many rounds of polynomially many parallel oracle calls.

The following theorem formalises the 'natural interpretation' of our result given in the intro.

Theorem 2.1.4 (Natural interpretation)
If $\mathsf{P}_{\|}^{\mathsf{ExactCount}} = \mathsf{P}_{\|}^{\mathsf{ApproxCount}}$ *then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$

This result becomes easy to show after Theorem 2.2.1, so we will prove it at the end of the next section.

2.2 PostBQP and PostBPP

An immediate consequence of the last section's main theorem is to the question of $\mathsf{PostBQP} \overset{?}{=} \mathsf{PostBPP}$, i.e. is a poly-time quantum computer with postselection equal to a classical computer with postselection? The best existing answer was given when Aaronson showed $\mathsf{PostBQP} = \mathsf{PP}$ [14], thus equality would imply $\mathsf{PH} = \mathsf{BPP}^{\mathsf{NP}}$ (a collapse to the third level). However, as $\mathsf{PostBPP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$ [20, 28] our theorem can be used to improve this collapse to the second level. Before we provide proof of this we formally define PostBQP and the operator $\mathsf{BP}\cdot$ which makes a complexity class probabilistic.

Definition 2.2.1
PostBQP *is the set of languages for which there exists a uniform family of polynomial width and polynomial depth quantum circuits $\{C_n\}_{n\geq 1}$ such that for any input x,*

(i) There is a non-zero probability of measuring the first qubit of $C_n |0\ldots 0\rangle |x\rangle$ to be $|1\rangle$.

(ii) If $x \in L$, conditioned on the first qubit being $|1\rangle$, the second qubit is $|1\rangle$ with probability at least 2/3.

(iii) If $x \notin L$, conditioned on the first qubit being $|1\rangle$, the second qubit is $|1\rangle$ with probability at most 1/3.

PostBPP can be defined similarly with classical circuits and additional input of random bits.

Definition 2.2.2
Let K be a complexity class. A language L, is in $\mathsf{BP}\cdot\mathsf{K}$ if there exists a language $\mathsf{A} \in \mathsf{K}$ and a polynomial p such that for all strings x.

$$Pr(r \in \{0,1\}^{p(|x|)} : x \in L \iff x\#r \in A) \geq 2/3$$

This covers the necessary definitions and we can proceed with the collapse result of interest.

Theorem 2.2.1
If $\mathsf{PostBQP} = \mathsf{PostBPP}$ then $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$ and the polynomial hierarchy collapses to the second level.

Proof. Assume $\mathsf{PostBQP} = \mathsf{PostBPP}$. By Toda $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$, by Aaronson [19] $\mathsf{PostBQP} = \mathsf{PP}$ and by $\mathsf{PostBPP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$ [20, 28]. This gives:

$$\mathsf{PP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}} \text{ and } \mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}} \subseteq \mathsf{P}^{\mathsf{BPP}^{\mathsf{NP}}} = \mathsf{BPP}^{\mathsf{NP}} = \mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}.$$

Therefore $\mathsf{P}^{\#\mathsf{P}} = \mathsf{PH}$.

At this point we recall an interesting lemma from Toda [25] to continue.

$$\mathsf{PP}^{\mathsf{PH}} \subseteq \mathsf{BP} \cdot \mathsf{PP}$$

Clearly $\mathsf{PH} \subseteq \mathsf{PP}^{\mathsf{PH}}$ and it can be shown that if $\mathsf{PP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then $\mathsf{BP} \cdot \mathsf{PP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$ (a full proof follows in lemma 2.2.1), therefore

$$\mathsf{PH} \subseteq \mathsf{BP} \cdot \mathsf{PP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}.$$

Since $\mathsf{PH} = \mathsf{P}^{\#\mathsf{P}}$ and $\mathsf{BPP}_{\|}^{\mathsf{NP}} \subseteq \mathsf{PH}$ we can conclude $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$. We now have the condition $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ so by theorem 2.1.1 we get $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$, the final result. □

Lemma 2.2.1
If PP *is in* $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ *then so is* $\mathsf{BP} \cdot \mathsf{PP}$

Proof. Fix some $L \in \mathsf{BP} \cdot \mathsf{PP}$. By the definition, there exists $A \in \mathsf{PP}$ which defines L. Assume PP is in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$, therefore A is in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$. Therefore L is in $\mathsf{BP} \cdot \mathsf{BPP}_{\|}^{\mathsf{NP}}$.

Note that majority-vote amplification can be used on $\mathsf{BP} \cdot \mathsf{PP}$ and $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ to decrease the probability of failure to less than $1/6$.

If we combine the random coin flips of both the $\mathsf{BP} \cdot \mathsf{PP}$ and BPP parts of the algorithm we get a single failure probability below $1/3$, which fits the definition of $\mathsf{BPP}_{\|}^{\mathsf{NP}}$. □

Theorem 2.2.1 provides a quick a simple proof of Theorem 2.1.4 (the natural interpretation of our result).

Proof of Theorem 2.1.4. By definition $\#\mathsf{P} = \mathsf{ExactCount}$. O'Donnell and Say show $\mathsf{P}_{\|}^{\mathsf{ApproxCount}} = \mathsf{PostBPP}$ [20]. Therefore we can rewrite the antecedent of the Theorem as $\mathsf{P}_{\|}^{\#\mathsf{P}} = \mathsf{PostBPP}$.

Any problem in PP can be solved with one $\#\mathsf{P}$ query, which can obviously be done in one parallel round of oracle calls, so $\mathsf{PP} \subseteq \mathsf{P}_{\|}^{\#\mathsf{P}}$. This gives $\mathsf{PP} = \mathsf{PostBPP}$ (as we already know $\mathsf{PostBPP} \subseteq \mathsf{PP}$ and by Theorem 2.2.1 $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$. □

2.3 Improved Hardness of BosonSampling, IQP, and DQC1

In the original work on the hardness of Boson Sampling [17] two cases were considered: *the exact case*, where the probability that the algorithm would sample a given element must be at least multiplicatively close to the target distribution, and *the approximate case*, which allowed additive error in the total variation distance between the sampled and target distribution. For the exact case, classical simulation implied the polynomial hierarchy collapsed to the third level. For the approximate case, a collapse to the 3rd level was achieved, but subject to additional assumptions (we refer to these as additional assumptions henceforth). This separation between multiplicative and additive error also applies to the work on instantaneous quantum polynomial-time sampling (IQP) [18] and sampling from 1 clean qubit circuits (DQC1) [29], albeit with different additional assumptions. It is important to notice that realistic quantum computers are believed not to be able to achieve multiplicative error in sampling, but can achieve additive error [18, 30]. So it is the approximate case that distinguishes quantum from classical computation.

Fujii [21] et al strengthened the collapse for the exact case to $\mathsf{PH} = \mathsf{AM} \cap \mathsf{coAM}$ (a collapse to the second level of the polynomial hierarchy) for BosonSampling, DQC1 and IQP (amongst others). However, for fundamental reasons, the proof did not extend to the approximate case.

In this section, we show that efficient classical sampling of the physically relevant *approximate case* would also collapse the polynomial hierarchy to its second level ($\mathsf{ZPP}^{\mathsf{NP}}$), conditional on the existing additional assumptions. In this sense, our work provides the strongest known separation between classical and quantum computations for sampling and functional problems. While our result can also be applied to the exact case it is not an improvement on Fujii's result, as it is known $\mathsf{AM} \cap \mathsf{coAM} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$, and this application is therefore not relevant.

To apply Theorem 2.1.1 and show a second level PH collapse we require the condition $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$, however, the proofs contained in [17, 18, 29] only show $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$. Fortunately, each of the proofs can be easily modified to only use parallel oracle calls, giving $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$. Formally proving this fact would require reproducing each proof in detail and would make this chapter excessively long. Instead of completely rewriting these proofs we will instead notice that each proof only uses the NP oracle to approximately count some post-selected quantity with Stockmeyers algorithm [31]. In the next lemma, we show that this computation can be done with parallel oracle calls. This will allow us to argue classical sampling of $\mathsf{BosonSampling}$, IQP or $\mathsf{DQC1}$ would imply $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\parallel}^{\mathsf{NP}}$ without formally reproducing each of the proofs.

Lemma 2.3.1 (Counting a postselected quantity)
Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a post selection criteria $h : \{0,1\}^n \to \{0,1\}$ let

$$p = \Pr_{x\in\{0,1\}^n}[f(x) = 1|h(x) = 1] = \frac{\sum_{x\in\{0,1\}^n} f(x)h(x)}{\sum_{x\in\{0,1\}^n} h(x)}.$$

Then for all $g \geq 1 + \frac{1}{poly(n)}$, there exists an $\mathsf{FBPP}_{\parallel}^{\mathsf{NP}^{f,h}}$ machine that approximates p to within a multiplicative factor of g.

The proof of lemma 2.3.1 is quite trivial once it is realised that Stockmeyer's approximate counting theorem can be done with only parallel oracle calls, which we capture in the next lemma.

Lemma 2.3.2 (Approximate counting in parallel)
Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, let

$$p = \Pr_{x\in\{0,1\}^n}[f(x) = 1] = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} f(x).$$

Then for all $g \geq 1 + \frac{1}{poly(n)}$, there exists an $\mathsf{FBPP}_{\parallel}^{\mathsf{NP}^{f}}$ machine that approximates p to within a multiplicative factor of g.

We will not provide all the steps of this lemma as it is a clear corollary of the version of the proof given by Valiant and Vazirani [32][4]. This theorem is also a consequence of the results of O'Donnel and Say [20], who show that approximate counting is in $\mathsf{PostBPP}$, and therefore in $\mathsf{BPP}_{\parallel}^{\mathsf{NP}}$.
We then proceed with the proof of lemma 2.3.1.

Proof of lemma 2.3.1. Fix a target error, g, from the assumption of the theorem there exists d such that $g > 1 + \frac{1}{n^d}$. By lemma 2.3.2 we can approximate $\sum_{x\in\{0,1\}^n} f(x)h(x)$ to a multiplicative factor of $g' = 1 + \frac{1}{3n^d}$ with high probability. Similarly we can approximate $\sum_{x\in\{0,1\}^n} h(x)$ to g' as well. Dividing the first sum by the second gives us p within a multiplicative factor of

$$g'^2 = 1 + \frac{2}{3n^d} + \frac{1}{9n^{2d}} \leq 1 + \frac{2}{3n^d} + \frac{1}{9n^d} = 1 + \frac{7}{9}\frac{1}{n^d} < 1 + \frac{1}{n^d}$$

[4]To see this, note that they can fix their random vectors at the start of their algorithm and check in parallel if 1, 2, up to n vectors are sufficient to empty the set.

with sufficiently high probability. Since each of these sums can be done in parallel and then divided for the final answer we can perform them in parallel. Given we only need to decrease the failure probability by a factor of 2 the final algorithm is in $\mathsf{FBPP}_{||}^{\mathsf{NP}^f}$. □

We will first apply the above logic for BosonSampling results. Aaronson and Arkhipov use one post-selection step in lemma 42 in [17], they then perform one approximate counting step on a quantity derived from this post-selection step to prove their main theorem. The structure of this proof fits the format of lemma 2.3.1, therefore we can state a parallelised version of their main theorem.

Theorem 2.3.1 (Aaronson and Arkhipov with parallel calls)
Let $\mathcal{D}_A$ be the probability distribution sampled by a boson computer A. Suppose there exists a classical algorithm C that takes as input a description of A as well as an error bound ε, and that samples from a probability distribution $\mathcal{D}'_A$ such that $\|\mathcal{D}'_A - \mathcal{D}_A\| \leq \varepsilon$ in poly$(|A|, 1/\varepsilon)$ time. Then the $|GPE|^2_\pm$ problem is solvable in $\mathsf{BPP}_{||}^{\mathsf{NP}}$.

The $|GPE|^2_\pm$ problem (defined in [17]) becomes $\#\mathsf{P}$ complete given two conjectures: The **permanent-of-Gaussians Conjecture** (PGC) and the **Permanent Anti-Concentration Conjecture** (PACC). These conjectures capture the belief that the permanent of Gaussian matrices does not concentrate close to zero and is hard to estimate, further justification of these conjectures is available in the original paper [17]. These are the *additional assumptions* we referred to earlier.

Theorem 2.3.2
Assume PACC and PGC hold. If there exists a classical algorithm that can sample the distribution of a boson computer with additive error, then $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$ and the polynomial hierachy collapse to the second level.

Next, we discuss how the same strategy applies to the IQP case. The proof provided by Bremner, Montanaro and Shepherd uses just Stockmeyer counting to prove their collapse to $\mathsf{BPP}^{\mathsf{NP}}$ in their theorem 1. By lemma 2.3.2 all NP calls in their algorithm can be done in parallel and we can convert their theorem 1 to a $\mathsf{BPP}_{||}^{\mathsf{NP}}$ result. For IQP we do not need to conjecture the concentration of some hard-problem. The result only rest on the conjectured average case hardness of approximately computing either the partition function of the Ising model *or* on a property of low-degree polynomials over finite fields (Conjectures 2 and 3 in [18]).

Theorem 2.3.3 (Improved IQP hardness [18])
Assume either above conjecture is true. If it is possible to classically sample from the output probability distribution of any IQP *circuit in polynomial time, up to an error of 1/192 in* l_1 *norm, then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$*. Hence the Polynomial Hierarchy would collapse to its second level.*

Finally, the same strategy can be applied to improve Morimae's result [29] on the hardness of the $\mathsf{DQC1}$ model for sampling [21] as it again depends on Stockmeyer counting. The necessary conjecture for Morimae's is an assumption directly about the average case hardness of the one clean qubit model.

Theorem 2.3.4 (Improved one clean qubit hardness [29])
Assuming Morimae's conjecture, if there exists a classical algorithm which can output samples from any one clean qubit machine with at most 1/36 error in the l_1 *norm then there is a* $\mathsf{ZPP}^{\mathsf{NP}}$ *algorithm to solve any problem in* $\mathsf{P}^{\#\mathsf{P}}$*. Hence the Polynomial Hierarchy would collapse to its second level.*

It seems likely that other results will fit the structure we give here, although we provide only these three results.

We finish this section by noting that the above results are in $\mathsf{SampBQP}$, so classical impossibility on any of these tasks would imply $\mathsf{SampBQP} \neq \mathsf{SampP}$, which would also imply a separation in the functional classes $\mathsf{FBQP} \neq \mathsf{FBPP}$ [14].

Theorem 2.3.5
If any of the sets of conjecture above hold and the polynomial hierarchy does not collapse to it's second level, then $\mathsf{FBQP} \neq \mathsf{FBPP}$ *and equivalently* $\mathsf{SampBQP} \neq \mathsf{SampP}$*.*

2.4 Conclusion and Open Problems

This chapter has shown that if $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then the polynomial hierarchy collapses to its second level. We have connected this result to approximate/exact counting and shown that it improves a number of results demonstrating the separation of quantum and classical computing. The natural next research question is 'how low can we go?'. Fujii et al [21] extended the result for the multiplicative error case to $\mathsf{AM} \cap \mathsf{coAM}$, perhaps this could be achieved. We have not used the fact that the $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm likely only needs two rounds of oracle calls. This could be an avenue to collapsing the hierarchy further.

This hierarchy collapse strengthens claims of quantum advantage but as these claims also rest on a number of additional assumptions (e.g. permanents-of-Gaussians conjectures), diminishing, removing or proving the other assumptions remain one of the central challenges of showing quantum advantage and proving $\mathsf{SampBQP} \neq \mathsf{SampP}$. Alternatively further work may reveal one of these assumptions to fall through, such as with XQUATH [30].

Our results offer other, more direct, extensions. Of particular interest is whether our main theorem relativises, like previous advantage results did [17]. Avoiding using the checkability of $\#\mathsf{P}$ may be key to proving relativisation as this is the only step of our proof that did not relativise.

Another open question is how Theorem 2.1.1 may be used for other non-quantum purposes, perhaps where approximate and exact counting are being compared. Alternatively, a research line we did not pursue is extending theorem 2.1.1. Extensions could be to other checkable rsr languages, perhaps PSPACE or EXP (although these may only be adaptively-rsr [23]), or to showing the equality holds for other elements of the polynomial hierarchy ($\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\Sigma_i} \overset{?}{\implies} \mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\Sigma_i}$).