Forrelation: A Problem That Optimally Separates Quantum from Classical Computing

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1145/2746539.2746547">http://dx.doi.org/10.1145/2746539.2746547</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Association for Computing Machinery (ACM)</td>
</tr>
<tr>
<td>Version</td>
<td>Author’s final manuscript</td>
</tr>
<tr>
<td>Citable link</td>
<td><a href="http://hdl.handle.net/1721.1/99662">http://hdl.handle.net/1721.1/99662</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-Noncommercial-Share Alike</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc-sa/4.0/">http://creativecommons.org/licenses/by-nc-sa/4.0/</a></td>
</tr>
</tbody>
</table>
ABSTRACT

We achieve essentially the largest possible separation between quantum and classical query complexities. We do so using a property-testing problem called FORRELATION, where one needs to decide whether one Boolean function is highly correlated with the Fourier transform of a second function. This problem can be solved using 1 quantum query, yet we show that any randomized algorithm needs $\Omega(\sqrt{N}/\log N)$ queries (improving an $\Omega(N^{1/4})$ lower bound of Aaronson). Conversely, we show that this 1 versus $\Omega(\sqrt{N})$ separation is optimal: indeed, any $t$-query quantum algorithm whatsoever can be simulated by an $O(N^{1-1/2t})$-query randomized algorithm. Thus, resolving an open question of Buhrman et al. from 2002, there is no partial Boolean function whose quantum query complexity is constant and whose randomized query complexity is linear. We conjecture that a natural generalization of FORRELATION achieves the optimal $t$ versus $\Omega(N^{1-1/2t})$ separation for all $t$. As a bonus, we show that this generalization is BQP-complete. This yields what’s arguably the simplest BQP-complete problem yet known, and gives a second sense in which FORRELATION “captures the maximum power of quantum computation.”

1. BACKGROUND

Since the work of Simon [15] and Shor [14] two decades ago, we have had powerful evidence that quantum computers can achieve exponential speedups over classical computers. Of course, for problems like FACTORING, these speedups are conjectural at present: we cannot rule out that a fast classical factoring algorithm might exist. But in the black-box model, which captures most known quantum algorithms, exponential and even larger speedups can be proved. We know, for example, that Period-Finding (a natural abstraction of the problem solved by Shor’s algorithm) is solvable with only $O(1)$ quantum queries, but requires $\tilde{O}(N)$ classical randomized queries, where $N$ is the number of input elements [8, 6, 12]. We also know that Simon’s Problem is solvable with $O(\log N)$ quantum queries, but requires $\Omega(\sqrt{N})$ classical queries; and that a similar separation holds for the Glued-Trees problem introduced by Childs et al. [7, 11].

To us, these results raise an extremely interesting question:

• “The Speedup Question.” Within the black-box model, just how large of a quantum speedup is possible? For example, could there be a function of $N$ bits with a quantum query complexity of 1, but a classical randomized query complexity of $\Omega(N)$?

One may object: once we know that exponential and even larger quantum speedups are possible in the black-box model, who cares about the exact limit? In our view, the central reason to study the Speedup Question is that doing so can help us better understand the nature of quantum speedups themselves. For example, can all exponential quantum speedups be seen as originating from a common cause? Is there a single problem or technique that captures the advantages of quantum over classical query complexity, in much the same way that random sampling could be said to capture the advantages of randomized over deterministic query complexity?

As far as we know, the Speedup Question was first posed by Buhrman et al. [5] around 2002, in their study of quantum property-testing. Specifically, Buhrman et al. asked whether there is any property of $N$-bit strings that exhibits a “maximal” separation: that is, one that requires $\Omega(N)$ queries to test classically, but only $O(1)$ quantumly. The best separation they could find, based on Simon’s problem, was “deficient” on both ends: it required $\Omega(\sqrt{N})$ queries to test classically, and $O(\log N \log \log N)$ quantumly.

Since then, there has been only sporadic progress on the Speedup Question. In 2009, Aaronson [1] introduced the FORRELATION problem—a problem that we will revisit in this paper—and showed that it was solvable with only 1
quantum query, but required $\Omega(N^{1/4})$ classical randomized queries. In 2010, Chakraborty et al. [6] argued that PERIOD-
FINDING gives a different example of an $O(1)$ versus $\Omega(N^{1/4})$
quantum/classical gap; there, however, we only get an $O(1)$-
query quantum algorithm if we allow non-Boolean queries.

Earlier, in 2001, de Beaudrap, Cleve, and Watrous [4] had
given what they described as a black-box problem that was
solvable with 1 quantum query, but that required $\Omega(N^{1/4})$
or $\Omega(\sqrt{N})$ classical randomized queries (depending on how
one defines the “input size” $N$). However, de Beaudrap et
al. were not working within the usual model of quantum
query complexity. Normally, one provides “black-box ac-
cess” to a function $f$, meaning that the quantum algorithm
can apply a unitary transformation that maps basis states of
the form $|x,y\rangle$ to basis states of the form $|x, y \oplus f(x)\rangle$ (or
|x⟩ to $(-1)^{f(x)}|x\rangle$, if $f$ is Boolean). By contrast, for their
separation, de Beaudrap et al. had to assume the ability to
map basis states of the form $|x,y\rangle$ to basis states of the form
|x, π(y + sx))⟩, for some unknown permutation $\pi$ and hidden
shift $s$.

2. OUR RESULTS

This paper has two main contributions—the largest quan-
tum black-box speedup yet known, and a proof that this
speedup is essentially optimal—as well as many smaller re-
lated contributions.

2.1 Maximal Quantum/Classical Separation

We undertake a detailed study of the FORRELATION
problem, which Aaronson [1] introduced for a different purpose
than the one that concerns us here (he was interested in an
oracle separation between BQP and the polynomial hierar-
chy). In FORRELATION, we are given access to two Boolean
functions $f, g : \{0, 1\}^n \to \{-1, 1\}$. We want to estimate the amount of correlation between $f$ and
the Fourier transform of $g$—that is, the quantity

$$\Phi_{f,g} := \frac{1}{2^{2n/2}} \sum_{x,y \in \{0,1\}^n} f(x) (-1)^{x \cdot y} g(y).$$

It is not hard to see that $|\Phi_{f,g}| \leq 1$ for all $f,g$. The problem
is to decide, say, whether $|\Phi_{f,g}| \leq \frac{1}{2}$ or $\Phi_{f,g} \geq \frac{3}{4}$; promised
that one of these is the case.3 Here and throughout this
paper, the “input size” is taken to be $N := 2^n$.

One can give (see Section 4.3) a quantum algorithm that
solves FORRELATION, with bounded probability of error, using
only 1 quantum query. Intuitively, however, the prop-
erty of $f$ and $g$ being “forrelated” (that is, having large $\Phi_{f,g}$
value) is an extremely global property, which should not be
apparent to a classical algorithm until it has queried a sig-
ificant fraction of the entire truth tables of $f$ and $g$. And
indeed, improving an $\Omega(N^{1/4})$ lower bound of Aaronson [1],
in Section 4.1 we show the following:

Theorem 1. Any classical randomized algorithm for FOR-
RELATION must make $O(\sqrt[4]{N})$ queries.

Theorem 1 yields the largest quantum versus classical sep-
aration yet known in the black-box model. As we show in
the full version, Theorem 1 also implies the largest property-
testing separation yet known—for with some work, one can recast
FORRELATION (or rather, its negation) as a property
that is testable with only 1 query quantumly, but that re-
quires $\Omega(\sqrt[4]{N})$ queries to test classically.

We deduce Theorem 1 as a consequence of a more gen-
eral result: namely, a lower bound on the classical query
complexity of a problem called GAUSSIAN DISTINGUISHING. Here
e we are given oracle access to a collection of $N'(0,1)$
real Gaussian random variables, $x_1, \ldots, x_M$. We are asked
to decide whether the variables are all independent, or alter-
atively, whether they lie in a known low-dimensional subspace of $\mathbb{R}^M$: one that induces a covariance of at most $\varepsilon$
between each pair of variables, while keeping each variable
an $N'(0,1)$ Gaussian individually. We show the following:

Theorem 2. GAUSSIAN DISTINGUISHING requires

$$\Omega \left( \frac{1}{\varepsilon^2 \log M/\varepsilon} \right)$$
classical randomized queries.

Theorem 2 is then simply a (discretized) special case of
Theorem 2, with $M = 2N$ and $\varepsilon = 1/\sqrt{N}$, the latter coming
from the inner product between a standard basis vector and
a Fourier basis vector. Beyond that, it seems to us that
Theorem 2 could have independent applications in statistics
and machine learning.

2.2 Proof of Optimality

We show that the quantum/classical query complexity
separation achieved by the FORRELATION problem is close
to the best possible. More generally:

Theorem 3. Let $Q$ be any quantum algorithm that makes
t = $O(1)$ queries to an $N$-bit string $X \in \{0,1\}^N$. Then
we can estimate $Pr[Q accepts X]$, to constant additive error
and with high probability, by making only $O(N^{1/4})$ clas-
sical randomized queries to $X$.4 Moreover, the randomized
queries are nonadaptive.

So for example, every 1-query quantum algorithm can be
simulated by an $O(\sqrt{N})$-query classical randomized
algorithm, every 2-query quantum algorithm can be simu-
lated by an $O(N^{3/4})$-query randomized algorithm, and so
on. Theorem 3 resolves the open problem of Buhrman et
al. [5] in the negative: it shows that there is no problem
property-testing or otherwise) with a constant versus linear
quantum/classical query complexity gap. Theorem 3 does
not rule out the possibility of an $O(\log N)$ versus $\tilde{O}(N)$
gap, and indeed, we conjecture that such a gap is possible.

Once again, we deduce Theorem 3 as a consequence of a
more general result, which might have independent applica-
tions to classical sublinear algorithms. Namely:

Theorem 4. Every degree-$k$ real polynomial $p : \{-1,1\}^N \to \mathbb{R}$
that is

4The reason for the condition $t = O(1)$ is that, in the
bound $O(N^{1/4} 2^t)$, the big-O hides a multiplicative factor of $\exp(t)$. Thus, we can obtain a nontrivial upper bound
on query complexity as long as $t = o(\sqrt{\log N})$.
(i) bounded in $[-1, 1]$ at every Boolean point, and

(ii) “block-multilinear” (that is, the variables can be partitioned into $k$ blocks, such that every monomial is the product of one variable from each block),

can be approximated to within $\pm \varepsilon$, with high probability, by nonadaptively querying only $O\left((N/\varepsilon^2)\left(\frac{1}{N}\right)^{1-1/k}\right)$ of the variables.

In the statement of Theorem 4, we strongly conjecture that condition (ii) can be removed. If so, then we would obtain a sublinear algorithm to estimate any bounded, constant-degree real polynomial. In the full version, we show that condition (ii) can indeed be removed in the special case $k = 2$.

### 2.2.1 $k$-fold Forrelation

Next, we study a natural generalization of Forrelation. In $k$-fold Forrelation, we are given access to $k$ Boolean functions $f_1, \ldots, f_k : \{0, 1\}^n \rightarrow \{-1, 1\}$, and we want to estimate the “twisted sum”

$$\Phi_{f_1, \ldots, f_k} := \frac{1}{2^{(k+1)n/2}} \sum_{x_1, \ldots, x_k \in \{0, 1\}^n} f_1(x_1)(-1)^{x_1 \cdot z_1} f_2(x_2)(-1)^{x_2 \cdot z_2} \cdots (-1)^{x_k \cdot z_k} f_k(x_k)$$

It is not hard to show that $|\Phi_{f_1, \ldots, f_k}| \leq 1$ for all $f_1, \ldots, f_k$. The problem is to decide, say, whether $|\Phi_{f_1, \ldots, f_k}| \leq \frac{1}{\sqrt{\omega}}$ or $|\Phi_{f_1, \ldots, f_k}| \geq \frac{2}{\sqrt{\omega}}$, promised that one of these is the case.

One can give (see Section 4.3) a quantum algorithm that solves $k$-fold Forrelation, with bounded error probability, using only $[k/2]$ quantum queries. In Section 4.4, we show conversely, that $k$-fold Forrelation “captures the full power of quantum computation”:

**Theorem 5.** If $f_1, \ldots, f_k$ are described explicitly (say, by circuits to compute them), and $k = \text{poly}(n)$, then $k$-fold Forrelation is a BQP-complete promise problem.

This gives us a particularly “self-contained” complete problem for quantum computation. Not only can one state the $k$-fold Forrelation problem without any notions from quantum mechanics, one does not need any nontrivial mathematical notions, like the condition number of a matrix or the Jones polynomial of a knot.

We conjecture, moreover, that $k$-fold Forrelation achieves the optimal $k/2$ versus $\Omega\left(N^{\frac{1}{1-k}}\right)$ quantum/classical query complexity separation for all even $k$. If so, then there are two senses in which $k$-fold Forrelation captures the power of quantum computation.

### 2.3 Other Results

In the full version of this paper, we also include several other results.

First, we study the largest possible quantum versus classical separations that are achievable for approximate sampling and relation problems. We show that there exists a sampling problem—namely, Fourier Sampling of a Boolean function—that is solvable with 1 quantum query, but requires $\Omega(\frac{N}{\log N})$ classical queries. By our previous results, this exceeds the largest quantum/classical gap that is possible for decision problems.

Second, we generalize our result that every 1-query quantum algorithm can be simulated using $O(\sqrt{N})$ randomized queries, to show that every bounded-degree real polynomial $p : \{-1, 1\}^N \rightarrow [-1, 1]$ can be estimated using $O(\sqrt{N})$ randomized queries. We conjecture that this can be generalized, to show that every bounded-degree $k$ real polynomial can be estimated using $O\left(N^{\frac{1}{1-1/k}}\right)$ randomized queries.

Third, we extend our $\Omega(\sqrt[3]{\frac{N}{\log N}})$ randomized lower bound for the Forrelation problem, to show a $\Omega(\sqrt[3]{\frac{N}{\log N}})$ lower bound for $k$-fold Forrelation for any $k \geq 2$. We conjecture that the right lower bound is $\Omega\left(N^{\frac{1}{1-1/k}}\right)$, but even generalizing our $\Omega(\sqrt[3]{\frac{N}{\log N}})$ lower bound to the $k$-fold case is nontrivial.

### 3. Query Complexity

Briefly, by the query complexity of an algorithm $A$, we mean the number of queries that $A$ makes to its input $z = (z_1, \ldots, z_N)$, maximized over all valid inputs $z$. The query complexity of a function $F$ is then the minimum query complexity of any algorithm $A$ (of a specified type—classical, quantum, etc.) that outputs $F(z)$, with bounded probability of error, given any valid input $z$.

One slightly unconventional choice that we make is to define “bounded probability of error” to mean “error probability at most $1/2 - \varepsilon$ for some constant $\epsilon > 0$” rather than “error probability at most $1/3$.” The reason is that we will be able to design a 1-query quantum algorithm that solves the Forrelation problem with error probability $2/5$, but not one that solves it with error probability $1/3$. Of course, one can make the error probability as small as one likes using amplification, but doing so increases the query complexity by a constant factor.

We assume throughout this paper that the input $z \in \{-1, 1\}^N$ is Boolean, and we typically work in the $\{-1, 1\}$ basis for convenience. In the classical setting, each query returns a single bit $z_i$, for some index $i \in [N]$ specified by $A$. In the quantum setting, each query performs a diagonal unitary transformation

$$|i, w\rangle \rightarrow z_i |i, w\rangle,$$

where $w$ represents “workspace qubits” that do not participate in the query.\(^5\) Between two queries, $A$ can apply any unitary transformation it likes that does not depend on $z$.

In this paper, the input $z = (z_1, \ldots, z_N)$ will typically consist of the truth tables of one or more Boolean functions: for example, $f, g : \{0, 1\}^n \rightarrow \{-1, 1\}$, or $f_1, \ldots, f_k : \{0, 1\}^n \rightarrow \{-1, 1\}$. Throughout, we use $n$ for the number of input bits that these Boolean functions take (which roughly corresponds to the number of qubits in a quantum algorithm), and we use $N = 2^n$ for the number of bits being queried in superposition. (Strictly speaking, we should set $N = k \cdot 2^n$, where $k$ is the number of Boolean functions. But this constant-factor difference will not matter for us.) Thus, for the purposes of query complexity, $N$ is the “input size,” in terms of which we state our upper and lower bounds.

---

\(^5\)If we are talking about a partial Boolean function, then a “valid” input is simply any input that satisfies the promise.\(^6\)For Boolean inputs $z$, this is well-known to be exactly equivalent to a different definition of a quantum query, wherein each basis state $|i, a, w\rangle$ gets mapped to $|i, a \oplus z_i, w\rangle$. Here $a$ represents a 1-qubit “answer register."
4. TECHNIQUES

4.1 Randomized Lower Bound

Proving that any randomized algorithm for FORRELATION requires $\Omega\left(\frac{\log N}{\log N} \right)$ queries is surprisingly involved. As we mentioned in Section 2.1, the first step, following the work of Aaronson [1], is to convert FORRELATION into an analogous problem involving real Gaussian variables. In REAL FORRELATION, we are given oracle access to two real functions $f, g : \{0, 1\}^n \rightarrow \mathbb{R}$, and are promised either that (i) every $f(x)$ and $g(y)$ value is an independent $\mathcal{N}(0, 1)$ Gaussian, or else (ii) every $f(x)$ value is an independent $\mathcal{N}(0, 1)$ Gaussian, while every $g(y)$ value equals $f(y)$ (i.e., the Fourier transform of $f$ evaluated at $y$). The problem is to decide which holds. Using a rounding reduction, we show that any query complexity lower bound for REAL FORRELATION implies the same lower bound for FORRELATION itself.

More formally, in the full version we prove the following.

**Theorem 6.** Suppose real functions $f, g : \{0, 1\}^n \rightarrow \mathbb{R}$ are drawn according to the measure (ii) above. Define Boolean functions $F, G : \{0, 1\}^n \rightarrow \{\pm 1\}$ by $F(x) := \text{sgn}(f(x))$ and $G(y) := \text{sgn}(g(y))$. Then

$$E_{f,g}[\Phi_{F,G}] = \frac{2}{\pi} + O\left(\frac{\log N}{N}\right).$$

Earlier, Aaronson [1, Theorem 9] proved a variant of Theorem 6, but with a badly suboptimal constant: he was only able to show that

$$E[\Phi_{F,G}] \geq \cos\left(2\arccos\sqrt{\frac{2}{\pi}}\right) - o(1) \approx 0.273,$$

compared to the exact value of $2/\pi \approx 0.637$ that we get here.

Theorem 6 has the following corollary, as we prove in the full version.

**Corollary 7.** Suppose there exists a $T$-query algorithm that solves FORRELATION with bounded error. Then there also exists an $O(T)$-query algorithm that solves REAL FORRELATION with bounded error.

So to prove a lower bound for FORRELATION, it suffices to prove the same lower bound for REAL FORRELATION. Furthermore, because the REAL FORRELATION problem is to distinguish two probability distributions, we can assume without loss of generality that any algorithm for the latter is deterministic.

Making the problem continuous allows us to adopt a geometric perspective. In this perspective, we are given oracle access to a real vector $v \in \mathbb{R}^{2N}$, whose $2N$ coordinates consist of all values $f(x)$ and all values $g(y)$ (recall that $N = 2^n$). We are trying to distinguish the case where $v$ is simply an $\mathcal{N}(0, 1)^{2N}$ Gaussian, from the case where $v$ is confined to an $N$-dimensional subspace of $\mathbb{R}^{2N}$—namely, the subspace defined by $g = f$. Now, suppose that values $f(x_1), \ldots, f(x_i)$ and $g(y_1), \ldots, g(y_u)$ have already been queried. Then we can straightforwardly calculate the Bayesian posterior probabilities for being in case (i) or case (ii). For case (i), the probability turns out to depend solely on the squared 2-norm of the vector of empirical data seen so far:

$$\Pr[\text{case (i)}] \propto \exp\left(-\frac{\Delta_i}{2}\right),$$

where

$$\Delta_i = f(x_1)^2 + \cdots + f(x_i)^2 + g(y_1)^2 + \cdots + g(y_u)^2.$$
his previous work, Aaronson [1] showed that the approximation holds until $\Omega(N^{1/4})$ queries are made. Indeed, he proved a stronger statement: even if the $x$’s and $y$’s are chosen nondeterministically, still $\Omega(N^{1/4})$ values must be revealed until we have a certificate showing that we are in case (i) or case (ii) with high probability.

To improve the lower bound from $\Omega(N^{1/4})$ to the optimal $\tilde{\Omega}(\sqrt{N})$, there are several hurdles to overcome.

Aaronson had assumed, conservatively, that the deviations from orthogonality all “pull in the same direction.” As a first step, we notice instead that the deviations follow an unbiased random walk, with some positive and others negative—the martingale property arising from the fact that the algorithm can control which $x$’s and $y$’s to query, but not the values of $f(x)$ and $g(y)$. We then use a Gaussian generalization of Azuma’s inequality to upper-bound the sum of the deviations. Doing this improves the lower bound from $\Omega(N^{1/4})$ to $\tilde{\Omega}(N^{1/3})$, but we then hit an apparent barrier.

In this work, we explain the $\Omega(N^{1/3})$ barrier, by exhibiting a “model problem” that is extremely similar to REAL FORRELATION (in particular, has exactly the same near-orthogonality property), yet is solvable with only $O(N^{1/3})$ queries, by exploiting adaptivity.

In more detail, recall the generalization of REAL FORRELATION that we call GAUSSIAN DISTINGUISHING. Here we are given a finite set $\mathcal{V}$ of unit vectors in $\mathbb{R}^N$, called “test vectors.” (In our case, $\mathcal{V}$ happens to equal $\{\{1\}, \{1\}, \ldots, \{N\}, \{N\}\}$.) In each step, we are allowed to pick any test vector $|v\rangle \in \mathcal{V}$ that has’t been picked in previous steps. We then “query” $|v\rangle$, getting back a real-valued response $a_v \in \mathbb{R}$. The problem is to distinguish the following two cases, with constant bias:

(i) Each $a_v$ is drawn independently from $N(0, 1)$.

(ii) Each $a_v$ equals $\langle \Psi | v \rangle$, where $|\Psi\rangle \in \mathbb{R}^N$ is a vector drawn from $N(0, 1)^N$ that is fixed throughout the algorithm.

In our case (REAL FORRELATION), we have $M = 2N$ and $\varepsilon = 1/\sqrt{N}$. Thus, one might hope one could prove a general lower bound on GAUSSIAN DISTINGUISHING, only depending on $\varepsilon$, of the form $\tilde{\Omega}(1/\varepsilon)$. Unfortunately, this is impossible: one does not have a $\tilde{\Omega}(1/\varepsilon)$ lower bound on query complexity independent of $M$, but at best a $\tilde{\Omega}(1/\varepsilon^2)$ lower bound. In the context of REAL FORRELATION, this means that, if the only thing we knew about $\mathcal{V}$ was that

[7]Here is the example that shows this: let $\{1\}, \ldots, |n\rangle$ be orthogonal unit vectors. Then for all $2^2$ strings $z = z_1 \cdots z_n \in \{-1, 1\}^n$, let $|w_z\rangle$ be a vector such that $\langle w_z | i \rangle = z_i/n^{3/2}$ for all $i \in [n]$, and also such that the projections of the $|w_i\rangle$’s onto the orthogonal complement of $\{1\}, \ldots, |n\rangle$ are all orthogonal to one another. Let $\mathcal{V} = \{\{1\}, \ldots, |n\rangle\} \cup \{|w_z\rangle : z \in \{-1, 1\}^n\}$. Then the inner product between any two distinct vectors in $\mathcal{V}$ is upper-bounded by $\varepsilon = 1/n^{3/2}$ in absolute value (the inner product between any two $|w_z\rangle$’s is at most $n/n^{3/2} = 1/n^{3/2}$). On the other hand, here is an algorithm that solves GAUSSIAN DISTINGUISHING using only $O(n) \leq 1/\varepsilon$ queries: first query $\{1\}, \ldots, |n\rangle$ to obtain $a_1, \ldots, a_n$. Let $|\varphi\rangle := a_1 |1\rangle + \cdots + a_n |n\rangle$. Next, find $n$ distinct vectors $|w_z\rangle$ that each have inner product $\Theta(n/n^{3/2}) = \Theta(1/\sqrt{n})$ with $|\varphi\rangle$ (such vectors can always $\langle \varphi | w_z \rangle \leq 1/\sqrt{N}$ for all distinct $|\varphi\rangle, |w_z\rangle \in \mathcal{V}$ (in particular, we had no upper bound on $\mathcal{V}$’s cardinality), then we could not hope to prove any lower bound better than $\Omega(N^{1/3})$.

However, we then break this barrier, by using the fact that, for REAL FORRELATION (but not for the model problem), the total number of vectors in $\mathcal{V}$ is only $N^{O(1)}$. This fact lets us use the Gaussian Azuma’s inequality a second time, to upper-bound not only the sum of all the deviations from orthogonality, but the individual deviations themselves. Implementing this yields a lower bound of $\tilde{\Omega}(N^{2/5})$: better than $\Omega(N^{1/3})$, but still not all the way up to $\tilde{\Omega}(\sqrt{N})$. However, we then notice that we can apply Azuma’s inequality recursively—once for each layer of the Gram-Schmidt orthogonalization process—to get better and better upper bounds on the deviations from orthogonality. Doing so gives us a sequence of lower bounds $\tilde{\Omega}(N^{1/7}), \tilde{\Omega}(N^{4/9}), \tilde{\Omega}(N^{5/11}),$ etc., with the ultimate limit of the process being $\tilde{\Omega}(\sqrt{N})$.

Our argument actually proves a general lower bound for GAUSSIAN DISTINGUISHING, which works whenever $|\mathcal{V}|$ is not too large, and every pair of vectors in $\mathcal{V}$ is sufficiently close to orthogonal. Here is our general result:

**Theorem 9.** Suppose $|\mathcal{V}| \leq M$, and $\langle \varphi | w_z \rangle \leq \varepsilon$ for all distinct vectors $|\varphi\rangle, |w_z\rangle \in \mathcal{V}$. Then any classical algorithm for GAUSSIAN DISTINGUISHING must make $\Omega \left( \frac{1/\varepsilon}{\log(M/\varepsilon)} \right)$ queries.

### 4.2 Randomized Upper Bound

Why did we have to work so hard to prove a $\tilde{\Omega}(\sqrt{N})$ lower bound on the randomized query complexity of FORRELATION? Our other main result provides one possible explanation: namely, we are here scraping up against the “ceiling” of the possible separations between randomized and quantum query complexity. In particular, any quantum algorithm that makes $1$ query to a Boolean input $X \in \{0, 1\}_n$, can be simulated by a randomized algorithm (in fact, a nonadaptive randomized algorithm) that makes $O(\sqrt{N})$ queries to $X$. More generally, any quantum algorithm that makes $t = O(1)$ queries to $X$, can be simulated by a nonadaptive randomized algorithm that makes $O(N^{1-1/t^2})$ queries to $X$.

The proof of this result consists of three steps. The first involves the simulation of quantum algorithms by low-degree polynomials. In 1998, Beals et al. [3] famously observed that, if a quantum algorithm makes $t$ queries to a Boolean input $X \in \{-1, 1\}^n$, then $p(X)$, the probability that the algorithm accepts $X$, can be written as a multilinear polynomial in $X$ of degree at most $2t$. We extend this result of Beals et al., in a way that might be of independent interest for quantum lower bounds. Namely, we observe that every $t$-query quantum algorithm gives rise, not merely to be found, so long as $|a_i| = \Omega(1)$ for $\Omega(n)$ values of $i$, and query all of them, letting $b_1, \ldots, b_n$ be the results. In case (i), we have $\mathbb{E}[b_1 + \cdots + b_n]$ and $\mathbb{V}[b_1 + \cdots + b_n] = n$. But in case (ii), we have $\mathbb{E}[b_1 + \cdots + b_n] = \Theta(\sqrt{n})$ and $\mathbb{V}[b_1 + \cdots + b_n] = O(n)$, allowing the two cases to be distinguished with constant bias.

In fact one can prove a $\tilde{\Omega}(N^{1/3})$ lower bound even under this restriction—and more generally, in the statement of Theorem 9, one can replace the lower bound $\Omega \left( \frac{1/\varepsilon}{\log(M/\varepsilon)} \right)$ by $\Omega \left( \frac{1/\varepsilon}{\log(M/\varepsilon)} \right)$, independent of $M$. We show how to do this in the full version.
a multilinear polynomial, but to a block-multilinear polynomial. By this, we mean a degree-2t polynomial q that takes as input 2t blocks of N variables each, and whose every monomial contains exactly one variable from each block. If we repeat the input X ∈ {−1, 1}^N across all 2t blocks, then q(X, . . . , X) represents the quantum algorithm’s acceptance probability on X. However, the key point is that q(Y) is bounded in [−1, 1] for any Boolean input Y ∈ {−1, 1}^{2tN}. More formally:

LEMMA 10. Let A be a quantum algorithm that makes t queries to a Boolean input x ∈ {−1, 1}^N. Then there exists a degree-2t block-multilinear polynomial p : R^{2tN} → R, with 2t blocks of N variables each, such that

(i) the probability that A accepts x equals p(x, . . . , x) (with x repeated 2t times), and

(ii) p(z) ∈ [−1, 1] for all z ∈ {−1, 1}^{2tN}.

PROOF. Assume for simplicity (and without loss of generality) that A involves real amplitudes only. For all j ∈ [t] and i ∈ [N], let x_{j,i} be the value of x_i that A’s oracle returns in response to its j^th query. Of course, in any “normal” run of A, we will have x_{j,i} = x_{j,j} for all j, j′; that is, the value of x_i will be consistent across all t queries. But it is perfectly legitimate to ask what happens if x_i changes from one query to the next. In any case, A will have some normalized final state, of the form

\[ \sum_{i,w} \alpha_{i,w} (x_{1,1}, \ldots, x_{t,N}) |i, w⟩. \]

Furthermore, following Beals et al. [3], it is easy to see that each amplitude \( \alpha_{i,w} \) can be written as a degree-t block-multilinear polynomial in the tN variables \( x_{1,1}, \ldots, x_{1,N} \), with one block of N variables, \( R_j = \{x_{j,1}, \ldots, x_{j,N}\} \), corresponding to each of the t queries. Next, for all j ∈ [t] and i ∈ [N], we create a second variable \( x_{j,i,j} \) which just like \( x_{j,i} \) represents the value of \( x_i \) that A’s oracle returns in response to its \( j^\text{th} \) query. Let Acc be the set of all accepting basis states, and consider the polynomial

\[ p(x_{1,1}, \ldots, x_{2t,N}) := \sum_{(i,w) \in \text{Acc}} \alpha_{i,w} (x_{1,1}, \ldots, x_{1,N}) \alpha_{i,w} (x_{t+1,1}, \ldots, x_{2t,N}). \]

By construction, \( p \) is a degree-2t block-multilinear polynomial in the 2tN variables \( x_{1,1}, \ldots, x_{2t,N} \), with one block of N variables, \( R_j = \{x_{j,1}, \ldots, x_{j,N}\} \), for each j ∈ [2t]. Furthermore, if we repeat the same input \( x \in \{-1,1\}^N \) across all 2t blocks, then

\[ p(x, \ldots, x) = \sum_{(i,w) \in \text{Acc}} \alpha_{i,w}^2 (x, \ldots, x) \]

is simply the probability that A accepts x. Finally, even if \( x_{1,1}, \ldots, x_{2t,N} \in \{-1,1\}^{2tN} \) is completely arbitrary, \( p \) still represents an inner product between two vectors. Since both of these vectors have norm at most 1, their inner product is bounded in [−1, 1].

This leads to a new complexity measure for Boolean functions \( f \): the block-multilinear approximate degree \( \bar{\deg}(f) \), which lower-bounds the quantum query complexity \( Q(f) \) just as \( \deg(f) \) does, but which might provide a tighter lower bound in some cases.

Once we have our quantum algorithm’s acceptance probability in the form of a block-multilinear polynomial \( p \), the second step is to preprocess \( p \), to make it easier to estimate using random sampling. The basic problem is that \( p \) might be highly “unbalanced”: certain variables might be hugely influential. Such variables are essential to query, but examining the form of \( p \) does not make it obvious which variables these are. To deal with this, we repeatedly perform an operation called “variable-splitting,” which consists of identifying an influential variable \( x_i \), then replacing every occurrence of \( x_i \) in \( p \) by \( \frac{1}{\sqrt{2}} (x_i + \ldots + x_i) \), where \( x_i, \ldots, x_i \) are newly-created variables set equal to \( x_i \). Observe that variable-splitting preserves the property that \( p \) is bounded in [−1, 1] at all Boolean points— for, regardless of how we set \( x_{j,l} \), the new value will simply equal the value of \( p \) with \( x_{j,l} \) set to \( \frac{x_{j,l} + \ldots + x_{j,l}}{\sqrt{2}} \), which in turn is a convex combination of \( p \) with \( x_{j,l} \) set to −1 and \( p \) with \( x_{j,l} \) set to 1. The point of doing this is that each \( x_{j,l} \) will be less influential in \( p \) than \( x_i \) itself was, thereby yielding a more balanced polynomial. We show that variable-splitting can achieve the desired balance by introducing at most \( \exp(t) \cdot O(N) \) new variables, which is linear in \( N \) for constant \( t \).

More formally, suppose

\[ p(x_{1,1}, \ldots, x_{k,N}) = \sum_{i_1, \ldots, i_k \in [N]} a_{i_1, \ldots, i_k} x_{1,i_1} \cdots x_{k,i_k} \]

is a bounded block-multilinear polynomial of degree \( k \). Set \( \delta := \frac{\varepsilon^2}{N} \). Then by repeatedly splitting variables, we wish to achieve the following requirement: for every nonempty set \( S \subseteq [k] \),

\[ \Lambda_S := \sum_{(i_j) \in S} \left( \sum_{(i_j) \in S} a_{i_1, \ldots, i_k}^2 \right) \leq \delta. \]

Our key lemma is the following.

LEMMA 11. Let \( S \subseteq [k] \) be nonempty. Then there is a sequence of variable-splittings that introduces at most \( 1/\delta \) new variables, and that produces a polynomial \( p' \) that satisfies \( \Lambda_S \leq \delta \).

PROOF. We start with the case \( S = [k] \). Then we have to ensure

\[ \sum_{i_1, \ldots, i_k \in [N]} a_{i_1, \ldots, i_k}^2 \leq \delta, \]

where \( a_{i_1, \ldots, i_k}^2 \) is the coefficient of \( x_{1,i_1} \cdots x_{k,i_k} \). Let

\[ V_i := \sum_{i_2, \ldots, i_k \in [N]} a_{i_1, i_2, \ldots, i_k}^2. \]

We now randomly set each \( x_{j,i} \) for \( j \geq 2 \), to be 1 or −1 with independent probability 1/2. Let

\[ X_i := \sum_{i_2, \ldots, i_k \in [N]} a_{i_1, i_2, \ldots, i_k} x_{2,i_2} \cdots x_{k,i_k}. \]

Then \( E[X_i^2] = V_i \). By the concavity of the square root function, this means \( E[|X_i|] \geq \sqrt{V_i} \). Hence

\[ E[|X_1| + \cdots + |X_k|] \geq \sqrt{V_1} + \cdots + \sqrt{V_k}. \]
If we set \( x_{i,j} = 1 \) whenever \( X_i \geq 0 \) and \( x_{i,j} = -1 \) otherwise, we get

\[
p(x_{1,1}, \ldots, x_{k,N}) = \sum_{i=1}^{N} x_{i,1}X_i = \sum_{i=1}^{N} |X_i|.
\]

Since \( p(x_{1,1}, \ldots, x_{k,N}) \) is bounded in \([-1, 1]\) at all Boolean points, this means that

\[
\sqrt{V_1} + \ldots + \sqrt{V_N} \leq 1.
\]

We now perform a sequence of variable-splittings. For each \( i \in [N] \), let \( m_i := \lceil \sqrt{V_i}/\delta \rceil \), so that

\[
\delta m_i \leq \sqrt{V_i} < \delta (m_i + 1).
\]

Then we split \( x_{i,j} \) into \( m_i + 1 \) variables. This replaces each term \( a_{i_1,\ldots,i_k} x_{i_1,1} \cdots x_{i_k,k} \) with \( m_i + 1 \) terms that each equal \( \frac{1}{m_i+1} a_{i_1,\ldots,i_k} x_{i_1,1} \cdots x_{i_k,k} \). Therefore, this variable-splitting reduces \( V_i \) to \( V_i / (m_i + 1) \), and decreases the sum (2) by \( \frac{m_i}{m_i+1} V_i \).

After we have performed such variable-splittings for each \( i \), the sum (2) becomes

\[
\sum_{i=1}^{N} \frac{V_i}{m_i+1} \leq \sum_{i=1}^{N} \frac{V_i}{\sqrt{V_i}/\delta} = \delta \left( \sqrt{V_1} + \ldots + \sqrt{V_N} \right) \leq \delta.
\]

The number of new variables that get introduced equals

\[
\sum_{i=1}^{N} m_i \leq \sum_{i=1}^{N} \sqrt{V_i}/\delta \leq \frac{1}{\delta}.
\]

The case \( S \subset [k] \) reduces to the case \( S = [k] \) in the following way. For typographical convenience, assume that \( S = [\ell] \) for some \( \ell \). Then substituting \( x_{i,j} = 1 \) for \( i > \ell \) transforms the polynomial \( p(x_{1,1}, \ldots, x_{k,N}) \) into the polynomial

\[
p'(x_{1,1}, \ldots, x_{\ell,N}) = \sum_{i_1,\ldots,i_{\ell}} \bar{a}_{i_1,\ldots,i_{\ell}} x_{i_1,1} \cdots x_{i_{\ell},\ell}
\]

where

\[
\bar{a}_{i_1,\ldots,i_{\ell}} := \sum_{i_{\ell+1,\ldots,i_{k}} \in [N]} a_{i_1,\ldots,i_{k}}.
\]

The statement of Lemma 11 now becomes

\[
\sum_{i_1,\ldots,i_{\ell} \in [N]} \bar{a}_{i_1,\ldots,i_{\ell}}^2 \leq \delta
\]

which can be achieved similarly to the previous case. \( \square \)

Lemma 11 has the following consequence.

**Corollary 12.** There is a sequence of variable-splittings that introduces at most \( 2^k/\delta \) new variables, and that produces a polynomial \( p' \) that satisfies \( \Lambda_S \leq \delta \) for every nonempty subset \( S \subseteq [k] \).

**Proof.** We simply apply the procedure of Lemma 11 once for each nonempty \( S \subseteq [k] \), in any order. Since there are \( 2^k \) possible choices for \( S \), and since each iteration adds at most \( 1/\delta \) variables, the total number of added variables is at most \( 2^k/\delta \). Furthermore, we claim that later iterations can never “undo” the effects of previous iterations. This is because, if we consider how the quantity

\[
\Lambda_S = \sum_{(ij) \in S} \left( \sum_{(ij) \in S} a_{i_1,\ldots,i_k} \right)^2
\]

is affected by variable-splittings applied to the variables in the \( j^{th} \) block, there are only two possibilities: if \( j \in S \) then \( \Lambda_S \) can decrease, while if \( j \not\in S \) then \( \Lambda_S \) remains unchanged. \( \square \)

We then apply Corollary 12 with the choice \( \delta = \varepsilon^2/N \). This introduces at most \( 2^k N/\varepsilon^2 = O(\varepsilon) \) new variables, and achieves \( \Lambda_S \leq \varepsilon^2/N \) for every \( S \).

From now on, we use \( n \) to denote the “new” number of variables per block, which is a constant factor greater than the “old” number \( N \).

Once we have a balanced polynomial \( q \), the last step is to give a query-efficient randomized algorithm to estimate its value. Our algorithm is the simplest one imaginable: we simply choose \( O(n^{1-1/21}) \) variables uniformly at random, query them, then form an estimate \( \tilde{q} \) of \( q \) by summing only those monomials all of whose variables were queried. In more detail, let

\[
b_{i_1,\ldots,i_k} := a_{i_1,\ldots,i_k} x_{i_1,1} \cdots x_{i_k,k}.
\]

Then

\[
q(x_{1,1}, \ldots, x_{k,n}) = \sum_{i_1,\ldots,i_k} b_{i_1,\ldots,i_k}.
\]

We can estimate this sum in the following way. For each \( i, j \) independently, let \( y_{i,j} \) be a \( (0,1) \)-valued random variable with \( \Pr[y_{i,j} = 1] = \frac{1}{n^{1/2}} \). We then take

\[
P := b_{i_1,\ldots,i_k} y_{1,i_1} \cdots y_{k,i_k}
\]

as our estimator.

Clearly, this is an unbiased estimator of \( q(x_{1,1}, \ldots, x_{k,n}) \), with expectation

\[
E[P] = \frac{q(x_{1,1}, \ldots, x_{k,n})}{n}.
\]

The result we need to prove is that \( \text{Var}[P] = O(\delta/n) \). If this is true, then performing \( O(1) \) repetitions of \( P \) allows us to estimate \( q(x_{1,1}, \ldots, x_{k,n}) \) with precision \( \sqrt{\frac{\delta}{n}} = \sqrt{\varepsilon/n} \cdot n = \varepsilon \).

This estimation can be carried out with \( O(n^{1-1/k}) \) queries because, to calculate \( P \), we only need the values of \( x_{i,j} \) with \( y_{i,j} = 1 \), and the number of such variables is \( O(n^{1-1/k}) \), with a very high probability.

The hard part, done in the full version, is of course to show that the variance is bounded. The proof of this makes heavy use of the balancedness property that was ensured by the preprocessing step.

Examining our estimation algorithm, an obvious question is whether it was essential that \( q \) be block-multilinear, or whether the algorithm could be extended to all bounded low-degree polynomials. In the full version, we take a first step toward answering that question, by giving an \( O(\sqrt{N}) \)-query randomized algorithm to estimate any bounded degree \( 2 \) polynomial in \( N \) Boolean variables. Once we drop block-multilinearity, our variable-splitting procedure no longer works, so we rely instead on Fourier-analytic results of Dinur et al. [9] to identify influential variables which we then split.
defined in Section 2.2.1. From this, it follows that we put. Observe that probability that this circuit accepts is exactly apply the following sequence of operations:

\[ f \] and "accept" (i.e., say that \( \Phi_{f_1, \ldots, f_k} \) itself: for example, that \( \Phi_{f_1, \ldots, f_k} \leq 1 \).

4.4 Other Results

BQP-Completeness. The proof that the \( k \)-fold Forrelation problem is PromiseBQP-complete is simple, once one has the main idea. The sum that defines \( k \)-fold Forrelation is, itself, an output amplitude for a particular kind of quantum circuit, which consists entirely of Hadamard and \( f \)-phase gates (i.e., gates that map \( |x \rangle \) to \((-1)^{f(x)} |x \rangle \) for some Boolean function \( f \)). Since the Hadamard and CCPHASE gates (corresponding to \( f(x, y,z) = yz \)) are known to be universal for quantum computation, one might think that our work is done. The difficulty is that the quantum circuit for \( k \)-fold Forrelation contains a Hadamard gate on every qubit, between every pair of \( f \)-phase gates, \( \text{whether we wanted Hadamards there or not} \). Thus, if we want to encode an arbitrary quantum circuit, then we need some way of canceling unwanted Hadamards, while leaving the wanted ones. We achieve this via a gadget construction.

In more detail, we need a gadget that lets us Hadamard some desired \textit{subset} of the qubits, \( S \subset [n] \), and not the qubits outside \( S \). For simplicity, suppose that \( |S| = 2 \), and let \( a \) and \( b \) be \( S \)'s elements. Our gadget, shown in Figure 2, consists of three CSIGN gates (i.e., gates that map \( |x,y \rangle \) to \((-1)^{xy} |x,y \rangle \)) on \( a \) and \( b \), sandwiched between Hadamard gates. Note that we can implement a CSIGN on \( a \) and \( b \) as \( U_{f_i} \), where \( f_i(z_1, \ldots, z_n) := (-1)^{z_ayz} \). Meanwhile, the Hadamard gates are just those that are automatically applied between each \( U_{f_i} \) and \( U_{f_{i+1}} \) in a quantum circuit for Forrelation. To see why the gadget works, consider the following identity:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{pmatrix}^3
= \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}.
\]

In particular, if we let \( C \) stand for CSIGN, \( H^{\otimes 2} \) for Hadamards on two qubits, and \( S \) for the 2-qubit SWAP gate, then

\[ H^{\otimes 2} CH^{\otimes 2} CH^{\otimes 2} C H^{\otimes 2} = SH^{\otimes 2}. \]

Contrast this with what happens if we apply the 2-qubit identity, \( I \), rather than \( C \), in the inner layers:

\[ H^{\otimes 2} IH^{\otimes 2} IH^{\otimes 2} = 1. \]

Thus, Hadamards get applied if \( C \) is chosen for the inner layers, but \textit{not} if \( I \) is chosen. So this gadget has the ef-
fect of Hadamarding $a$ and $b$, while not Hadamarding the other qubits in the circuit. Now, the gadget also has the unintended side effect of swapping $a$ and $b$. But since we know this is going to happen, we can keep track of it by simply swapping the labels of $a$ and $b$ whenever the gadget is applied.

Indeed, examining the construction, we can derive a stronger consequence. Define a depth-$d$ quantum circuit as one where the gates are organized into $d$ sequential layers, with the gates within each layer all commuting with one another. Now, given a depth-$d$ quantum circuit $Q$ over the basis $\{H, CC\text{SIGN}\}$ (where $H$ is Hadamard), let QSim$_d$ be the problem of deciding whether the circuit’s accepting amplitude $A$ satisfies $A \geq \frac{1}{\sqrt{d}}$ or $|A| \leq \frac{1}{\log d}$, promised that one of those is the case. Then in the full version, we prove the following:

**Theorem 14.** QSim$_d$ is polynomial-time reducible to explicit $(2d+1)$-fold Forrelation. (Moreover, the functions $f_1, \ldots, f_{2d+1}$ produced by the reduction all have the form $f_i(x) = (-1)^{p(x)}$, where $p$ is a degree-3 polynomial in the input bits.)

So for example, we find that explicit log-$n$-fold Forrelation is a complete promise problem for PromiseBQNC$^1$; the class of problems that captures what can be done using log-depth quantum circuits.

**Separation for Sampling Problems.** To achieve a 1 versus $\Omega(N/\log N)$ quantum/classical query complexity separation for a sampling problem, we consider Fourier Sampling: the problem, given oracle access to a Boolean function $f : \{0, 1\}^n \to \{-1, 1\}$, of outputting a string $y \in \{0, 1\}^n$ with probability approximately equal to $f(y)^2$. This problem is trivially solvable with 1 quantum query, but proving a $\Omega(N/\log N)$ classical lower bound takes a few pages of work. The basic idea is to concentrate on the probability of a single string—say, $y = 0^n$—being output. Using a binomial calculation, we show that this probability cannot depend on $f$’s truth table in the appropriate way unless $\Omega(N/\log N)$ function values are queried.

**Lower Bound for k-Fold Forrelation.** Once we have a $\Omega(N/\log N)$ randomized lower bound for Forrelation, one might think it would be trivial to prove the same lower bound for k-fold Forrelation: just reduce one to the other! However, Forrelation does not embed in any clear way as a subproblem of k-fold Forrelation. On the other hand, given an instance of k-fold Forrelation, suppose we “give away for free” the complete truth tables of all but two of the functions. In that case, we show that the induced subproblem on the remaining two functions is an instance of Gaussian Distinctness to which, with high probability, our lower bound techniques can be applied. Pursuing this idea leads to our $\Omega(N/\log^{1/2} N)$ lower bound on the randomized query complexity of k-fold Forrelation, for all $k \geq 2$.

**Property-Testing Separation.** To turn our quantum versus classical separation for the Forrelation problem into a property-testing separation, we need to prove two interesting statements. The first is that function pairs $(f, g)$ that are far in Hamming distance from the set of all pairs with low correlation, actually have high correlation. The second is that “generic” function pairs $(f, g)$ and $(f', g')$ that have small Hamming distance from one another, are close in their correlation values as well. In fact, in the full version, we prove both of these statements for the general case of k-fold Forrelation.

5. **Discussion**

To summarize, this paper proves the largest separation between classical and quantum query complexities yet known, and it also proves that that separation is in some sense optimal. These results put us in a position to pose an intriguing open question:

Among all the problems that admit a superpolynomial quantum speedup, is there any whose classical randomized query complexity is $\gg \sqrt{N}$?

Strikingly, if we look at the known problems with superpolynomial quantum speedups, for every one of them the classical randomized lower bound seems to hit a “ceiling” at $\sqrt{N}$. Thus, Simon’s Problem has quantum query complexity $O(\log N)$ and randomized query complexity $\Theta(\sqrt{N})$; the Glued-Trees problem of Childs et al. [7] has quantum query complexity $\log^{O(1)}(N)$ and randomized query complexity $\Theta(\sqrt{N})$; and Forrelation has quantum query complexity $1$ and randomized query complexity $\Theta(\sqrt{N})$.

If we insist on making the randomized query complexity $\Omega(N^{1/2+c})$, for some $c > 0$, and then try to minimize the quantum query complexity, then the best thing we know how to do is to take the OR of $N^c$ independent instances of Forrelation, each of size $N^{1-2c}$. This gives us a problem whose quantum query complexity is $\Theta(N^c)$, and whose classical randomized query complexity is $\Theta(N^{1/2+c})$. Of course, this is not an exponential separation.

In this paper, we gave a candidate for a problem that breaks the “$\sqrt{N}$ barrier”: namely, k-fold Forrelation. Indeed, we conjecture that k-fold Forrelation achieves the

---

9The randomized lower bound for Glued-Trees proved by Childs et al. [7] was only $\Omega(N^{1/4})$. However, Fenner and Zhang [11] improved the lower bound to $\Omega(N^{1/3})$; and if we allow a success probability that is merely (say) $1/3$, rather than exponentially small, then their bound can be improved further, to $\Omega(\sqrt{N})$. In the other direction, we are indebted to Shalev Ben-David for proving that Glued-Trees can be solved deterministically using only $O(\sqrt{N} \log N)$ queries (or $O(\sqrt{N} \log^2 N)$, if the queries are required to be Boolean). For his proof, see http://cstheory.stackexchange.com/questions/25279/the-randomized-query-complexity-of-the-conjoined-trees-

10Here the upper bound comes from combining Grover’s algorithm with the Forrelation algorithm: the “naïve” way of doing this would produce an additional log $N$ factor for error reduction, but it is well-known that that log factor can be eliminated [13]. Meanwhile, the lower bound comes from the optimality of Grover’s algorithm.

11Here the upper bound comes from simply taking the best randomized Forrelation algorithm, which uses $O(\sqrt{N} \log N)$ queries, and running it $N^{2c}$ times, with an additional log $N$ factor for error reduction. Meanwhile, the lower bound comes from combining this paper’s $\Omega(\sqrt{N} / \log N)$ lower bound for Forrelation, with a general result stating that the randomized query complexity of OR ($f, \ldots, f$), the OR of $k$ disjoint copies of a function $f$, is $\Omega(k)$ times the query complexity of a single copy. This result can be proved by adapting ideas from a direct product theorem for randomized query complexity given by Drucker [10] (we thank A. Drucker, personal communication).
optimal separation for all $k = O(1)$, requiring $\tilde{\Omega}(N^{1-1/k})$ classical randomized queries but only $[k/2]$ quantum queries.\footnote{And perhaps $k$-fold \textsc{Forrelation} continues to give optimal separations, all the way up to $k = O(\log N)$.} Proving this conjecture is an enticing problem. Unfortunately, $k$-fold \textsc{Forrelation} becomes extremely hard to analyze when $k > 2$, because we can no longer view the functions $f_1, \ldots, f_k$ as confined to a low-dimensional subspace: now we have to view them as confined to a low-dimensional manifold, which is defined by degree-$(k-1)$ polynomials. As such, we can no longer compute posterior probabilities by simply appealing to the rotational invariance of the Gaussian measure, which made our lives easier in the $k = 2$ case. Instead we need to calculate integrals over a nonlinear manifold.

Short of proving our conjecture about $k$-fold \textsc{Forrelation}, it would of course be nice to find any partial Boolean function whose quantum query complexity is polylog $N$, and whose randomized query complexity is $N^{1/2+\Omega(1)}$.

Another problem we leave is to generalize our $O(N^{1-1/k})$ randomized estimation algorithm from block-multilinear polynomials to arbitrary bounded polynomials of degree $k$. As we said, in the full version we achieve this in the special case $k = 2$. Achieving it for arbitrary $k$ seems likely to require generalizing the machinery of Dinur et al. \cite{dinur}.

A third problem concerns the notion of block-multilinear approximate degree, $\text{bmdeg}(f)$, that we introduced to prove Theorem 4. Is there any asymptotic separation between $\text{bmdeg}(f)$ and ordinary approximate degree? What about a separation between $\text{bmdeg}(f)$ and quantum query complexity, as Ambainis \cite{ambainis} showed between $\tilde{\deg}(f)$ and quantum query complexity?

A fourth, more open-ended problem is whether there are any applications of \textsc{Forrelation}, in the same sense that factoring and discrete log provide "applications" of Shor’s period-finding problem. Concretely, are there any situations where one has two efficiently-computable Boolean functions $f, g : \{0,1\}^n \to \{-1,1\}$ (described, for example, by circuits), one wants to estimate how forrelated they are, and the structure of $f$ and $g$ does not provide a fast classical way to do this?

Here are five other open problems:

First, can we tighten the lower bound on the randomized query complexity of \textsc{Forrelation} from $\Omega(\sqrt{\log N})$ to $\Omega(\sqrt{N})$, or give an $O(\sqrt{N})$ upper bound?

Second, can we generalize our results from Boolean to non-Boolean functions?

Third, what are the possible greatest quantum versus classical query complexity separations for \textit{sampling} problems? Is an $O(1)$ versus $\Omega(N)$ separation possible in this case? Also, what separations are possible for search or relation problems? (For our results on these questions, see the full version.)

Fourth, while there exists a 1-query quantum algorithm that solves \textsc{Forrelation} with bounded error probability, the error probability we are able to achieve is about 0.4—more than the customary 1/3. If we want (say) a 1 versus $N^{O(1)}$ quantum versus classical query complexity separation, then how small can the quantum algorithm’s error be?

Fifth, in the full version we show that being “unforrelated”—that is, having $\Phi_{f,g} \leq \frac{1}{100}$—behaves nicely as a property-testing problem. But it would be interesting to show the same for being forrelated.

6. ACKNOWLEDGMENTS

We are grateful to Ronald de Wolf for early discussions, Andy Drucker for discussions about the randomized query complexity of OR$(f, \ldots, f)$, Shalev Ben-David and Sean Hallgren for discussions about the glued-trees problem, Saeed Mehrabian for a numerical calculation, and Man-Hong Yung for catching a calculation error in an earlier draft.

7. REFERENCES

\footnotesize


