The Table Maker's Quantum Search

Benjamin Morrison, Stefanos Kourtis

IEEE ARITH 2026

2026-07-01

UdS Université de Sherbrooke

NON TRIVIAL

INSTITUT QUANTIQUE UdS

Outline

HTR Problem

In a given binade $B_e = [2^e, 2^{e+1})$, what precision p is required to round a transcendental function $f(x)$ correctly to precision n, for all $x \in B_e$?

Exact: $\tilde{\mathcal{O}}\left(2^{\frac{2n}{3}}\right)$ (Lefevre 2005)

Heuristic: $\tilde{\mathcal{O}}\left(2^{\frac{n}{2}}\right)$ (Stehlé 2006)

Or if reordering is needed, (Hanrot 2007)

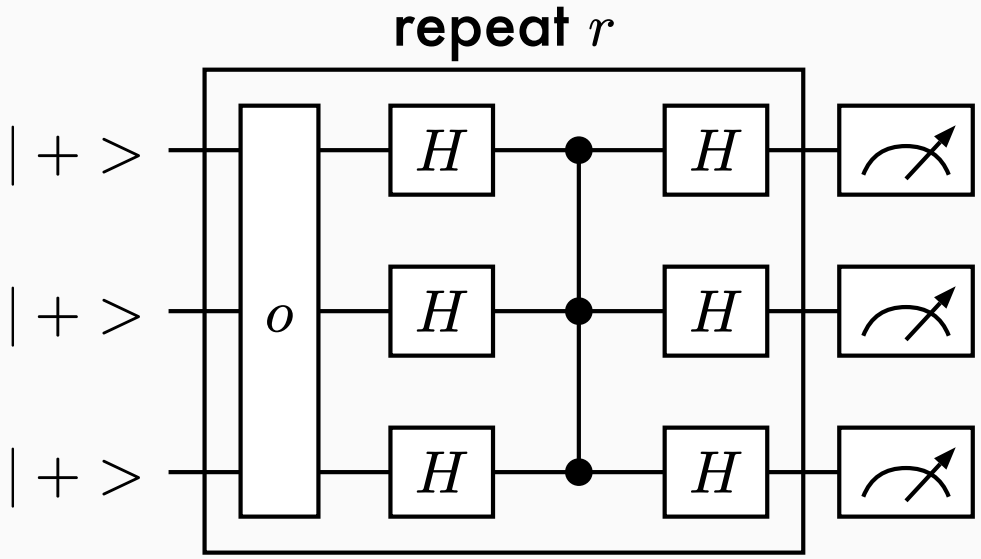
Exact: $\tilde{\mathcal{O}}\left(2^{\frac{4n}{5}}\right)$

Heuristic: $\tilde{\mathcal{O}}\left(2^{\sim.676n}\right)$

Can we get exact results with the performance of the heuristics?

Grover Search Algorithm

- Quantum computers search $\mathcal{O}(N)$ elements in $\tilde{\mathcal{O}}\left(\sqrt{N}\right)$ time. (Grover 1996)
- $|S|$ elements marked by an "oracle" function: $o(\vec{x}) = \begin{cases} -1 \text{ if } \vec{x} \in S \\ +1 \text{ otherwise} \end{cases}$
- Oracle implementation: reversible circuit: $|\vec{x}> \rightarrow o(x)|\vec{x}>$



Grover Search Algorithm

Grover search succeeds with probability

$$\sin^2\left(\frac{2r+1}{2}\theta\right)$$

after r iterations, where

$$\cos\left(\frac{\theta}{2}\right) = \sqrt{\frac{N-|S|}{N}}$$

- Normally assumes |S| is known and finds a marked element.
- Instead: test whether $|S| > 0$.
- Since $|S|$ is unknown, θ is unknown, and thus optimal r is unknowm.
- Iterating over $\mathcal{O}(\log N)$ values of r is required. (Boyer 1996)

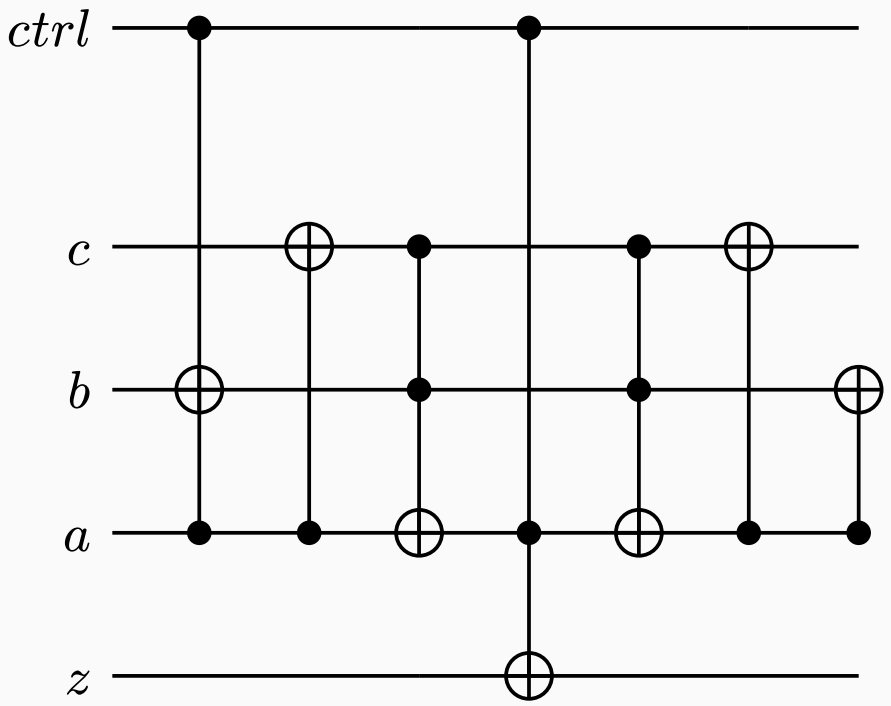
Reversible Arithmetic

- Quantum circuits are unitary.
- Classical subset of unitary operations implements reversible logic.

$\mathrm{X} : A \to \neg A \qquad \mathrm{CX} : A, B \to A, A \oplus B$

$\mathrm{Toffoli} : A, B, C \to A, B, (A \wedge B) \oplus C$

- Toffoli gate is expensive, X and CX are negligible.
- Reversible classical logic is universal. (Toffoli 1980)
- Ripple-carry addition circuits can be made reversible.
- Karatsuba-type circuits efficiently implement multiplication. (Parent 2018)
- Transcendental functions evaluated by these operations.



Algorithm Structure

Algorithm

- Pick candidate upper bound r for hardness to round to l bits.
- Start testing whether $\frac{r-l}{2}$ bits is sufficient to round all cases correctly.
- Quantum Grover search repeatedly calls oracle circuit and diffusion circuit.
  - Oracle circuit calculates $f(x)$ to upper bound precision.
  - Oracle circuit checks bit pattern for bad rounding case and sets flag.
- Set r to $\frac{r-l}{2}$ if no bad cases are found, and otherwise l to $\frac{r-l}{2}$.
- Repeat binary search to find hardness to round.

Membership Oracle

Algorithm

- Computes $f(x)$ to precision p given x with precision $n < p$.
- Polynomial-time classical computation; asymptotically $\mathcal{O}(M(p)\log p)$. (Brent 2010)
- Convert to reversible logic with polynomial overhead. (Bennett 1989)
- Output only the significand of $f(x)$.
- Checks if the last $p - n + 1$ bits of the significand match a bad rounding pattern.
- Multi-controlled phase gate implemented at $\mathcal{O}(p)$ Toffoli cost.
- Uncompute all values to leave the auxiliary register in $|0>$ state.
- Include constant-size machinery to handle overflow/underflow and outputs exactly equal to a float value.

Complexity

Algorithm

- Oracle circuit size is $\mathrm{poly}(p)$.
- $\mathcal{O}\left(2^{\frac{n}{2}}\right)$ Grover Search iterations detects an empty set with probability $\mathcal{O}(1)$.
- $\mathcal{O}\left(\log \delta'^{-1}\right)$ repetitions boosts the success probability to $1-\delta'$.
- Suppose $p_{\max} = \mathcal{O}(\mathrm{poly}(n))$ is an upper bound for the htr.
- Binary search requires $\lfloor \log p_{\max} \rfloor - 1$ iterations to find htr.
- For overall success probability δ, set $\delta' \leq \frac{\delta}{\lfloor \log p_{\max} \rfloor - 1}$.
- The overall runtime is $\tilde{\mathcal{O}}\left(2^{\frac{n}{2}} \log\left(\frac{1}{\delta}\right)\right)$.

Practicality?

- How hard is it to round $\sin$ and $\cos$ in `binary64`?
- Frontier of classical possibility! (Lefèvre 2026)
- Can quantum methods compare?
- Oracle: calculate $\sin(x)$ to 106 bits precision with binary splitting. (Brent 1976)
- Partition $x \bmod 2\pi$ into eight sections y_j, calculate $\sin$ and $\cos$ of each, and combine. (Brent 2010)

$$S_j = \sin(y_j)C_{j+1} + \cos(y_j)S_{j+1}$$
$$C_j = \cos(y_j)C_{j+1} - \sin(y_j)S_{j+1}$$

- No section requires more than 16 terms to evaluate, and the largest four sections are evaluated with a single term each.
- Cost per section no more than $3M(128) + 6M(64) + 12M(32) + 24M(16)$.
- Latency roughly $M(128) + M(64) + M(32) + M(16)$.

Reversible Multiplication

Practicality?

- Mixed-width integer multiplication dominates cost.
- 8-bit schoolbook multiplication costs 232 Toffoli gates.
- Karatsuba-style circuits cost $M(2^k) = 3M(2^{k-1}) + 12 \cdot 2^k$. (Parent 2018)

Circuit	$M(128)$	$M(64)$	$M(32)$	$M(16)$
Schoolbook (TOF count)	65152	16192	4000	976
Karatsuba (TOF count)	31272	9912	3048	888
Karatsuba (depth/latency)	3112	1576	808	424

- Can we do better in reversible logic?

Resource Estimation

Practicality?

- Total Toffoli depth per oracle call: 2.8×10^4.
- Total Toffoli count per oracle call: 2.3×10^6.
- Not possible on any current or near-future quantum computer!
- Optimistically: fault tolerant Toffoli time $\sim$ present-day measurement time $\sim 100\text{ns}$. (Spring 2025)
- Time per oracle call: $\sim 10\text{ms}$.
- Single binade search: 7.45×10^7 oracle calls, hundreds of hours.
- Full space search: 2.39×10^9 oracle calls, thousands of hours.
- For "quantum advantage" on binary64, need multiple orders of magnitude improvements! (Lefèvre 2026)
- Or expect crossovers in the tens of thousands of hours / 80+ bits regime?

Conclusion

- Asymptotic speedup from pure quantum search.
- Mostly exhaustive search; potentially improvable by the past 25 years of ARITH results.
- First known asymptotic quantum advantage on a computer arithmetic task.
- Fundamental, or can classical algorithms gain the same advantage?
- Connection to factoring?
- Can better arithmetic circuits make implementation more realistic? In particular, is binary splitting wise?