iQuHack 2026 (Superquantum Challenge): Leveraging Modularity for Twelve Unitary Circuit Decompositions

Adam Godel[1], Yebin Song[1], Nico Jackson[1], Travis Meyer[1], and Timothy Wright[1]

[1]Boston University, Boston, Massachusetts 02215, USA

February 1, 2026

Abstract

We consider the Clifford + T gate decomposition of twelve unitary gate operators, implementing modular gate implementations based on known near-optimal decompositions of the $R_z(\theta)$ gate with some precision ϵ. We also examine the phase polynomial's ability to lead to reductions in the T gate count based on coefficient parity or raw value depending on the gate restrictions.

1 Introduction

The development and optimization of fault-tolerant quantum circuits has been at the forefront of quantum research in recent years. This weekend at iQuHack, Superquantum tasked us with implementing efficient approximate quantum circuit implementations of unitary gate operators using the Clifford + T gate set, where the latter is by far the most computationally difficult gate to execute and functions as a description of the "quantumness" of a circuit.[1]

Our approach to implement these unitaries varied widely depending on the operator. Some have symmetries that can be exploited, while others are completely random and require brute-force optimization. Furthermore, there is a subjective tradeoff between minimizing the number of T gates and minimizing the distance between the approximate and the exact unitaries. In this writeup, we present our approaches to implement all twelve operators tasked in our challenge, considering how our implementations can be varied in both of these optimization regimes.

We will begin by establishing some background on quantum computing and circuit theory, then continue by discussing each of our implementations in detail, including our experimental results, and conclude with our reflection and possible future work on these problems.

[1]Our code is available on GitHub at the following link: `https://github.com/adam-godel/2026-Superquantum`.

2 Background

A *two-qubit quantum state* is represented as

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \tag{1}$$

where $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$. A quantum state can be acted upon with *quantum gates*, which are represented as unitary matrices $A \in U(2^n)$ for n qubits such that $|\psi\rangle \mapsto A|\psi\rangle$, which is a valid n-qubit quantum state.

The following quantum gates are known as the *Clifford gates* [1]:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\ S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}\ CX = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix} \tag{2}$$

When combined with the T gate, defined as

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}, \tag{3}$$

they form a *universal gate set* [2], meaning that any unitary operation can be approximated to arbitrary precision using circuits built with just these four gates.

Moreover, the *Gottesman-Knill theorem* [3] shows that the Clifford gates can be simulated in polynomial time on a classical computer. This reduces the "hardness" of a quantum circuit to be described in the number of T gates it contains.

3 Our Implementations

We now go through each of the twelve unitaries we implemented during the weekend, and the optimizations we made to leverage both low T gate counts and higher accuracy. We describe the implementations in order, although the difficulty in constructing each implementation varied widely.

3.1 Controlled-Y Gate

The *controlled-Y gate* is defined as

$$CY = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix}. \tag{4}$$

To implement this gate, we consider that $(I \otimes S)CX(I \otimes S^\dagger) = CY$, where the $\dagger$ operator denotes the conjugate transpose. Since we can implement this gate using only Clifford gates, there is no tradeoff—we get high accuracy and no T gate cost.

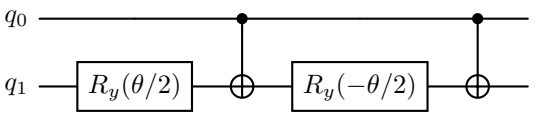
3.2 Controlled-$R_y(\pi/7)$ Gate

The *controlled-R_y gate* is defined as

$$CR_y(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta/2) & -\sin(\theta/2) \\ 0 & 0 & \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}. \tag{5}$$

We want to consider the case where $\theta = \pi/7$, which does not map cleanly onto a Clifford + T gate set. Therefore, we have to consider approximations. This is when we developed our central strategy for creating these implementations: ***modularity*** of gates.

Suppose we had a R_y gate in our toolbox. If this were the case, we could represent $CR_y(\theta)$ with the following circuit:



(6)

We know that this is true by considering the state of q_1 based on the state of q_0. We know that $CR_y(\theta)$ should apply $R_y(\theta)$ if $q_0 = |1\rangle$ and do nothing if $q_0 = |0\rangle$. In Eq. 7, we can see that if $q_0 = |0\rangle$, nothing is done to q_1, while if $q_0 = |1\rangle$, we can see that a $R_y(\theta)$ gate is applied to q_1.

The problem now reduces to an implementation of $R_y(\theta)$. We can observe that since $SXS^\dagger = Y$ and $HZH = X$, we have $R_y(\theta) = SHR_z(\theta)HS^\dagger$, i.e. we can represent $R_y(\theta)$ in terms of just $R_z(\theta)$ and Clifford gates. It is easy to see that the same is true for $R_x(\theta)$.

We implement a Clifford + T gate approximation of $R_z(\theta)$ using the `gridsynth` package [4], which takes in an angle θ and a precision ϵ and computes a decomposition for $R_z(\theta)$ with a T-gate count within $O(\log(\log(1/\epsilon)))$ of optimal.

This allows us to modularly use the R_x, R_y, and R_z gates, noting that they are the most expensive operation, so we should aim to minimize their usage.

3.3 Exponential of a Pauli String

We next tackle the implementation of $\exp(i\frac{\pi}{7}ZZ)$. We can observe that

$$\exp(i\theta ZZ) = CX(I \otimes \exp(i\theta Z))CX. \tag{7}$$

Consider the R_z gate, which is defined as

$$R_z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}. \tag{8}$$

It is not hard to see that $\exp(i\theta Z) = R_z(-2\theta)$. Therefore, we can implement this unitary similarly to the previous one but using only one R_z gate.

3.4 Exponential of a Hamiltonian

Consider the unitary operator $\exp(i\frac{\pi}{7}H_1)$, where $H_1 = XX+YY$. We implement each summand separately in a construction is similar to above. We implement $\exp(i\frac{\pi}{7}XX)$ by taking the previous circuit and conjugating with H, and we implement $\exp(i\frac{\pi}{7}YY)$ by conjugating with HS and $S^\dagger H$ on both qubits. This allows us to implement the exponential of this Hamiltonian using only two R_z gates.

3.5 A Curiously Simple Hamiltonian

Suppose we consider a similar Hamiltonian to the one above, but with a slightly different construction. We consider $\exp(i\frac{\pi}{4}H_2)$, where $H_2 = XX + YY + ZZ$. Can we implement this operator more efficiently than the previous ones?

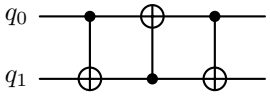
It turns out that we can, using the SWAP gate, which is defined as

$$\text{SWAP} = \frac{1}{2}(II + XX + YY + ZZ). \tag{9}$$

Using the SWAP gate, we can rewrite our exponential as

$$\begin{aligned} \exp\left(i\frac{\pi}{4}H_2\right) &= \exp\left(i\frac{\pi}{4}(2\text{SWAP} - I)\right) \\ &= e^{-i\frac{\pi}{4}} \exp\left(i\frac{\pi}{2}\text{SWAP}\right) \\ &= e^{i\frac{\pi}{4}}\text{SWAP} \end{aligned} \tag{10}$$

where the coefficient is a global phase that we can discard. Therefore, the Hamiltonian exponential is equivalent to the SWAP gate, which we can easily write using three CNOTs by the XOR swap algorithm:



(11)

This form makes the Hamiltonian trivial to simulate.

3.6 Two-Qubit Transverse Field Ising Model

We construct this exponential operator, defined as $\exp(i\frac{\pi}{7}H_3)$, where $H_3 = XX + ZI + IZ$. We construct the XX term in the same way as described in Sec. 3.4. For the other two terms, we simply apply $R_z(-2\theta)$ to each qubit.

3.7 State Preparation

We now discuss the implementation of an arbitrary state preparation

$$\begin{aligned} |00\rangle \mapsto\ & (0.1061479384 - 0.679641467i)|00\rangle \\ & +(-0.3622775887 - 0.453613136i)|01\rangle \\ & +(0.2614190429 + 0.0445330969i)|10\rangle \\ & +(0.3276449279 - 0.1101628411i)|11\rangle. \end{aligned} \tag{12}$$

This task differs from the previous ones since it is totally random. There is no structure we can exploit to systematically minimize the number of T gates we're using.

Consider how this unitary operator might look. For a unitary operator U where

$$U|00\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \tag{13}$$

its structure in the computational basis would be reminiscent of the following:

$$U = \begin{bmatrix} 0.1061479384 - 0.679641467i & * & * & * \\ -0.3622775887 - 0.453613136i & * & * & * \\ 0.2614190429 + 0.0445330969i & * & * & * \\ 0.3276449279 - 0.1101628411i & * & * & * \end{bmatrix} \tag{14}$$

where the $*$ symbols depict "free variables" that we don't care about the specific values of. The key observation is that for this matrix to be unitary, these columns must form an orthonormal basis, and the specific values of these free variables will greatly affect the number of T gates.

Therefore, our protocol is the following: we first generate a complete unitary using the Gram-Schmidt process, encoding randomness on many different diagonal matrices that are applied to each sample unitary [5]. We then test each candidate unitary over

many different values of ϵ. To balance the tradeoff, we start with a fidelity threshold compared to the expected state, and once that is hit, we then test over ϵ to minimize the number of T gates.

More precisely, we transpile the circuit into rotation gates using `qiskit`, and then decompose each one with a different most optimized ϵ value by comparing individually to the fidelity of the state with that operator applied as well as its individual T gate count.

This approach allows us to get a highly optimized circuit without incorporating any sort of structural symmetry; our approach works on fully random state preparation instances.

3.8 A Structured Unitary

We aim to apply the unitary

$$U_1 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}. \tag{15}$$

Notice that this unitary is the two-qubit quantum Fourier transform [6]. Since it only requires z rotations that are multiples of $\frac{\pi}{4}$, we do not need any rotation gates. Our implementation uses only three T gates.

3.9 Another Structured Unitary

We aim to apply the unitary

$$U_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2}+\frac{i}{2} & \frac{1}{2}+\frac{i}{2} \\ 0 & i & 0 & 0 \\ 0 & 0 & -\frac{1}{2}+\frac{i}{2} & -\frac{1}{2}-\frac{i}{2} \end{bmatrix}. \tag{16}$$

We can observe that this unitary, without a SWAP gate applied to it, is reminiscent of a controlled operation. Specifically, we can identify it as the controlled-$R_x\left(\frac{\pi}{4}\right)$ gate with some local phase changes, specifically $S \otimes TS$ gates before the SWAP. Since $\theta = \frac{\pi}{4}$, we can apply this directly as a T gate. Ultimately, our implementation uses four T gates in total.

3.10 Random Unitary

In the case of a random unitary operator, we can just use the exact same protocol as in Sec. 3.7 but without the generation of the unitary, since we are already restricted to what the unitary needs to be. Similarly to the state preparation problem, this approach gives us an efficient circuit without exploiting any structure relating to the unitary.

3.11 Four-Qubit Diagonal Unitary

We want to implement the four-qubit unitary U such that $U|x\rangle = e^{i\frac{\pi}{8}\varphi(x)}|x\rangle$, where $\varphi(x)$ is some four-bit boolean function mod 8 known as the *phase polynomial*.

The key observation is that the phase polynomial fully encodes the unitary, and the number of odd coefficients will correspond with the number of T gates. We first came up with some implementation for $\varphi(x)$, regardless of its optimality, and implemented it as a circuit.

We then used the `rmsynth` package [7] to optimize the phase polynomial, aiming to minimize both the number of T gates and the T gate depth. Specifically, we just used the documentation defaults: a decoder of `rpa`, an effort level of `3`, and a policy lambda of `5`.

From there, we get a dictionary representing the optimized phase polynomial, which we just implement as a quantum circuit as we have for the others.

This gives us a T gate count of 4 with nearly perfect accuracy.

3.12 n-Qubit Pauli List

We also implemented the bonus unitary, which is defined as

$$U = \prod_{j=1}^{m} \exp\left(-i\frac{\pi}{8}k_j P_j\right) \tag{17}$$

for Pauli strings P_j with coefficients k_j. Since all P_j commute pairwise, we can perform exact diagonalization using Clifford gates to get each Pauli gate for each qubit in the right basis.

We end up with the non-Clifford operator being in the form $e^{i\frac{\pi}{8}\varphi(x)}$, just like our construction in Sec. 3.11. From there, we take the same approach, but

excluding S gates since they are prohibited according to the rules of the challenge.

Given that constraint, we want to optimize for the lowest absolute coefficients as opposed to the lowest number of odd coefficients. We were able to get a T gate count of 287; due to time constraints, we weren't really able to optimize this unitary construction in the same way we did for the others.

4 Simulations

Given the intrinsic tradeoff between T-gate count and approximation accuracy, our research investigated different optimization regimes for various target operators. For unitaries $2, 3, 4, 6, 7$, and 10, we utilized a modular strategy where the circuit's total T-count was primarily determined by the precision (ϵ) of our $R_z(\theta)$ gate decompositions.

4.1 Unitary 2

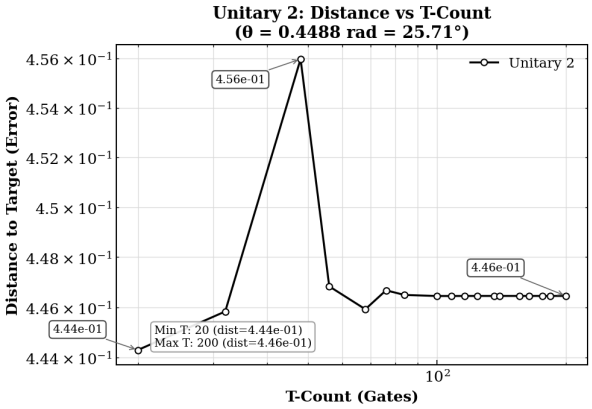


Figure 1: T-count vs. distance tradeoff for Unitary 2.

Figure 1 depicts varying ϵ values for Unitary 2. The plot for Unitary 2 exhibits a non-monotonic "spike" in error, where the distance reaches a maximum of 4.56×10^{-1} at mid-range T-counts due to local optimization failures. As the gate budget increases to 200, the approximation stabilizes at a persistent plateau of 4.46×10^{-1}, indicating that further T-gates do not significantly improve fidelity for this specific rotation.

4.2 Unitary 3

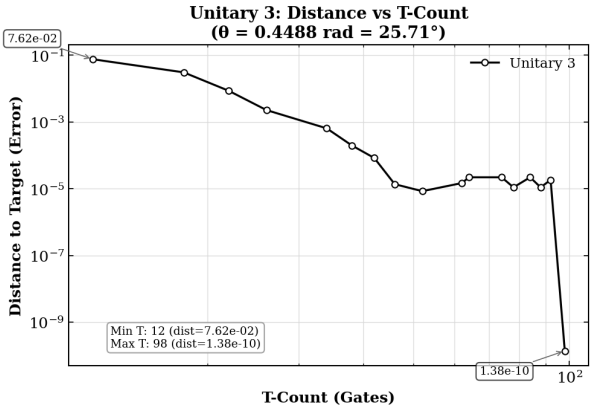


Figure 2: T-count vs. distance tradeoff for Unitary 3.

Figure 2 depicts varying ϵ values for Unitary 3. Unitary 3 demonstrates a successful exponential decrease in error, with the distance dropping from 7.62×10^{-2} down to a high-precision value of 1.38×10^{-10}. The profile features a distinct "breakthrough" near a T-count of 98, where the algorithm identifies an optimal Clifford + T sequence that dramatically reduces the residual error.

4.3 Unitary 4

Figure 3 depicts varying ϵ values for Unitary 4. This operator follows a steady downward trend punctuated by a significant error plateau between T-counts of 100 and 180. Upon reaching a budget of 196 gates, the distance collapses from roughly 10^{-5} to 1.96×10^{-10}, proving that higher T-gate thresholds are sometimes necessary to unlock high-precision approximations.

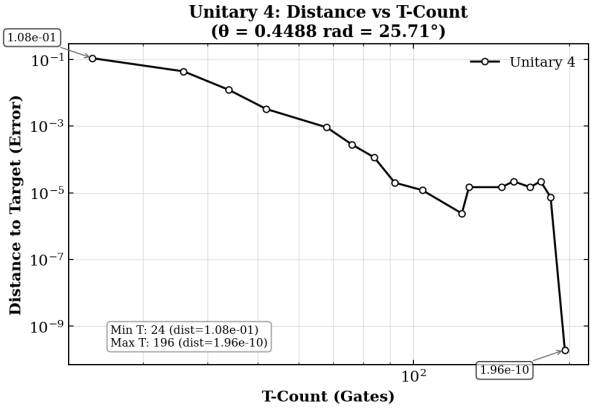


Figure 3: T-count vs. distance tradeoff for Unitary 4.

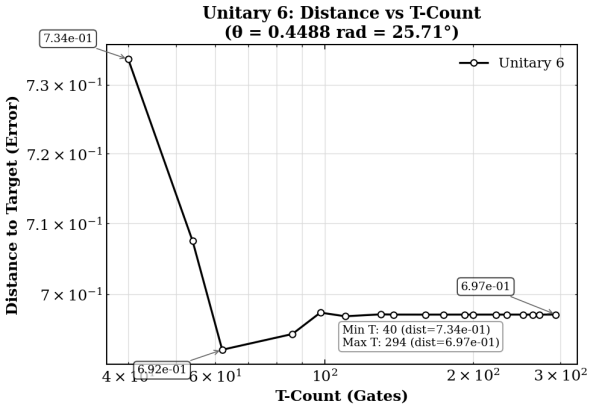


Figure 4: T-count vs. distance tradeoff for Unitary 6.

Variant	T-count	Distance
YYY	294	0.5090
YYN	196	0.8959
YNY	196	0.8959
YNN	98	1.132
NYY	196	0.9714
NYN	98	1.252
NNY	98	1.252
NNN	0	1.446

Table 1: Comparison of T-count and normalized distance for different Clifford variants. A "Y" marker indicates that a rotation gate is used, while an "N" marker indicates that a Clifford approximation is used ($S^\dagger$ gates in this case).

4.4 Unitary 6

Figure 4 depicts varying ϵ values for Unitary 6. Unitary 6 shows a rapid initial error reduction that quickly hits a hard ceiling, with the distance settling at 6.97×10^{-1} despite increasing the T-count to nearly 300. This behavior suggests a structural mismatch where the specific target unitary remains mathematically distant from the gate sequences reachable within the current search depth.

As an example, we decided to also investigate using pure Clifford gate approximations for Unitary 6. Table 1 shows the results, where an "N" marker indicates that an $S^\dagger$ gate was used instead of an $R_z\left(-\frac{2\pi}{7}\right)$ gate. The results indicate that including all of the rotation gates gives the lowest distance at the expense of having the highest T-count, while as we use more Clifford approximations, the accuracy decreases as the T-count decreases, which is what we expect.

4.5 Unitary 7

Figure 5 depicts varying ϵ values for Unitary 7. For Unitary 7, the analysis shifted from distance to Fidelity, targeting an extremely narrow high-precision band between 0.9999990 and 1.0. The plot reveals a stochastic, oscillatory pattern between T-counts of 490 and 592, characteristic of the randomized search protocol used to satisfy the strict ≥ 0.999999 fidelity threshold. This high resource cost and non-linear be-

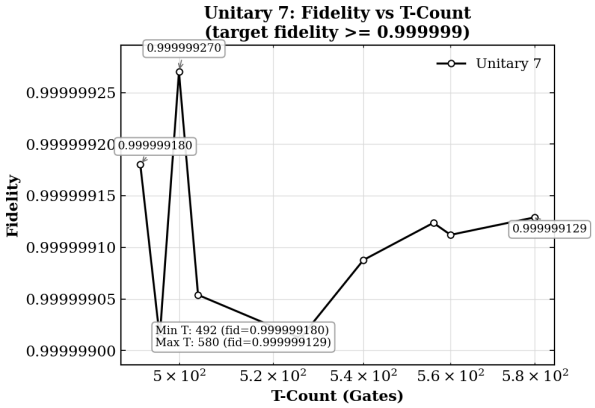


Figure 5: T-count vs. distance tradeoff for Unitary 7.

havior are typical for "totally random" state preparations that lack exploitable structural symmetries.

4.6 Unitary 10

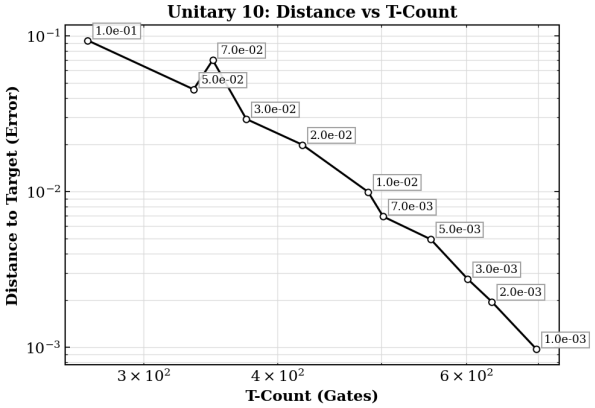


Figure 6: T-count vs. distance tradeoff for Unitary 10.

Figure 6 depicts varying *epsilon* values for Unitary 10. This unitary requires a much larger resource budget, scaling from 250 to over 700 gates to achieve a precision of 1.0×10^{-3}. The log-log linear trend, despite a minor local instability at 350 gates, confirms that this "random" operator demands high gate density to achieve even moderate fidelity compared to more structured gates.

5 Reflection & Conclusions

In this project, we showed that while some unitaries can be easily decomposed into all Clifford gates, the ones that can't have a tradeoff between T gate counts and accuracy. There is no objectively best configuration—ultimately, it depends on your use case and the ability of your hardware to simulate T gates.

For the later unitaries, we also identified the power of the phase polynomial to map many different classes of unitaries into Clifford and R_z gates, providing the modularity that really made our implementations possible and efficient.

We really enjoyed this project as a whole, and we are interested to explore the power of Clifford + T decomposition as well as the `rmsynth` package and this problem's connection to the Reed-Muller code further. We definitely think there is still room for improvement in our implementations, particularly Unitary 12, and are excited to continue exploring better and even more modular approaches past the hackathon deadline!

References

[1] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Phys. Rev. A*, 57:127–137, Jan 1998.

[2] Adam Sawicki, Lorenzo Mattioli, and Zoltán Zimborás. Universality verification for a set of quantum gates. *Phys. Rev. A*, 105:052602, May 2022.

[3] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, 70:052328, Nov 2004.

[4] Neil J. Ross and Peter Selinger. Optimal ancilla-free clifford+t approximation of z-rotations. *Quantum Info. Comput.*, 16(11–12):901–953, September 2016.

[5] Kaining Zhang, Min-Hsiu Hsieh, Liu Liu, and Dacheng Tao. Quantum gram-schmidt processes and their application to efficient state read-out for quantum algorithms. *Physical Review Research*, 3:043095, 2021.

[6] Neil J. Ross and Peter Selinger. Optimal ancilla-free clifford+t approximation of Z-rotations. 2014.

[7] Superquantum. rmsynth: High-performance clifford+t circuit optimizer using phase-polynomial methods and punctured reed–muller decoding, 2026. GitHub repository.