Cost-optimal single qubit gate synthesis in the Clifford hierarchy

Gary J. Mooney,1,∗ Charles D. Hill,1,2,† and Lloyd C.L. Hollenberg1,‡

1School of Physics, University of Melbourne, VIC, Parkville, 3010, Australia.
2School of Mathematics and Statistics, University of Melbourne, VIC, Parkville, 3010, Australia.

(Dated: May 22, 2020)

For universal quantum computation, a major challenge to overcome for practical implementation is the large amount of resources required for fault-tolerant quantum information processing. An important aspect is implementing arbitrary unitary operators built from logical gates within the quantum error correction code. A synthesis algorithm can be used to approximate any unitary gate up to arbitrary precision by assembling sequences of logical gates chosen from a small set of universal gates, which are fault-tolerantly performable while encoded in a quantum error-correction code. However, current procedures do not yet support individual assignment of base gate cost values and many do not support extended sets of universal base gates. We study cost-optimal sequences synthesised from sets of base gates which include Clifford gates and \( Z \)-rotation gates from higher orders of the Clifford hierarchy, which can be performed fault-tolerantly on error-correction codes using magic state distillation protocols. The individual costs assigned are the average numbers of raw (i.e. physical level) magic states required to implement the gates fault-tolerantly. By including the \( Z \)-rotation gates from the fourth order of the Clifford hierarchy as base gates in addition to the canonical Clifford+\( T \) gates, we find that the average cost decreases by up to 30%. The gate synthesis algorithm introduced in this work, based on Dijkstra’s algorithm, generates cost-optimal sequences for single-qubit target gates and supports arbitrary universal sets of single-qubit base gates with individually assigned cost values. In addition, we develop an analytic model to estimate the proportion of sets of \( Z \)-rotation gates from higher orders of the Clifford hierarchy among gates within sequences approximating random target gates, which can be used to estimate each order’s effectiveness for the purpose of gate synthesis.

I. INTRODUCTION

Quantum computing has the potential to solve many real-world problems by using significantly fewer physical resources and computation time than the best known classical algorithms. The quantum algorithms for these problems are implemented using deep quantum circuits. Thus to reliably implement these circuits, qubits within the devices require long coherence times and high precision control. Current systems consist of physical qubits that are too noisy for large scale computation. Error-correction schemes provide the ability to overcome this hurdle by entangling clusters of physical qubits in such a way that they collectively encode the information into more robust logical qubits. In principle, when physical qubits have error-rates below the error threshold of the error-correction scheme, logical qubits within the code can be made arbitrarily robust using increasing numbers of qubits. A particular error-correction scheme with relatively high physical error threshold of approximately 1% is the surface code, which is implemented over a nearest-neighbor two-dimensional physical layout, making it one of the most realistically implementable schemes [1–4]. In this work, we analyse the resource costs for gate synthesis, which is used to fault-tolerantly implement arbitrary unitary gates in error-correction codes.

The surface code, among other high-threshold codes, is limited to a small set of Clifford gates over logical qubits that can be performed with relative ease. A procedure called magic state distillation can be used to perform a wider range of non-Clifford gates fault-tolerantly, such as the \( T := R_z(\pi/4) \) gate (up to global phase), which cannot be produced using only Clifford gates [5, 6]. Initially, raw magic states are surgically injected into the code and with the aid of state distillation procedures, a number of raw magic states are consumed to produce a smaller number of more robust magic states. In principle, the procedures can be recursively applied to obtain states with arbitrarily low noise, although requiring large amounts of physical resources. These purified magic states can then be consumed to fault-tolerantly perform corresponding gates using quantum teleportation circuits. Distillation procedures only exist for a subset of gates, in order to implement arbitrary unitary gates, the Solovay-Kitaev (SK) theorem can be used. The SK theorem states that a universal set of \( n \)-qubit gates generate a group dense in \( SU(2^n) \) (Special Unitary), and

∗gmooey@student.unimelb.edu.au
†cdhill@unimelb.edu.au
‡lloydch@unimelb.edu.au
the set fills SU(2^n) relatively quickly. Hence single-qubit base gates that form a universal set can be multiplied in sequence to approximate any single-qubit gate to arbitrary precision [7].

A frequently used set of single-qubit universal base gates for fault-tolerant quantum computation are the Clifford+T gates, where the Clifford gates are relatively cheap to apply while the T gate requires a considerable amount of resources due to the magic state distillation procedure. This set of gates and how they can be used to synthesise arbitrary single-qubit gates is a well studied topic within the quantum compilation literature. Gate synthesis algorithms, besides brute-force [8], began with the Solovay-Kitaev algorithm [9]. It initially searches for a base sequence that roughly approximates a target gate and then uses a recursive strategy to append other base sequences in such a way that the new sequence approximates a gate that is closer to the target gate with distance reducing efficiently with the number of iterations. It is compatible with arbitrary single-qubit universal gate sets, provided that they include each gate’s adjoint. The SK algorithm has room for optimisation with respect to lengths of resulting gate sequences since the recursive process generates strings of disjoint subsequences which are only individually optimised, rather than optimising over the entire sequence. In 2008, Matsumoto and Amano [10] developed a normal form for sequences of Clifford+T gates that produces unique elements in SU(2). Shortly after, Bocharov and Svore [11] introduced their canonical form which extends the normal form by instead producing unique elements in PSU(2) (Projective Special Unitary) which more concisely describes the space of all physical single-qubit gates by ignoring global phase. This normal form can be used to enumerate length optimal sequences of Clifford+T base gates which produce distinct gates, considerably reducing the size of the sequence configuration space for search algorithms (although still growing exponentially with respect to sequence length).

More recently, there has been significant progress on developing direct synthesis methods which are not based on search. For target single-qubit unitary gates that can be exactly produced by Clifford+T gate sequences, a method was developed that optimally and efficiently finds these exact sequences directly [12]. This was later used as a subroutine in algorithms for optimal synthesis of arbitrary single-qubit Z-rotations [13, 14]. Direct Clifford+T gate base gate synthesis methods for Z-rotations have since been generalised to Clifford+cyclotomic (Z-rotation by π/n) sets of base gates [15] and sets derived from totally definite quaternion algebras [16]. For arbitrary single-qubit rotations (not necessarily Z-rotations) there has been a number of other approaches developed, such as a randomised algorithm that uses the distribution of primes [17], asymptotically optimal synthesis using ancilla qubits [18], and probabilistic quantum circuits with fallback [19].

It is common within the quantum compilation literature for synthesis algorithms to optimise sequences based on minimising the total number of gates that require magic state injection. This measure is well-suited to the Clifford+T set of base gates which are standard for gate synthesis algorithms, since the T gate and its adjoint are the only gates with a significantly higher cost than the Clifford gates. However, distillation procedures exist for performing alternative gates to the T gate that vary in implementation cost. Examples of such gates are found within the Clifford hierarchy, which is an infinite discrete set of gates that are universal and can be performed on certain error-correcting codes fault-tolerantly [20]. The resource cost of distillation varies between orders of the hierarchy. Thus to accurately cost optimise sequences from such sets of gates, the cost of each individual base gate must be considered. As a starting point for estimating resource costs of individual gates, Campbell and O’Gorman calculated the average number of raw magic states required to distil and implement subsets of gates belonging to the Clifford hierarchy for various levels of precision [21]. Although other factors relating to physical resources are important to consider such as qubit count and circuit depth, the number of raw magic states can serve as a first order approximation to the cost of implementing fault-tolerant logical gates.

In this work, we introduce an algorithm that generates a database of all cost-optimal sequences below a chosen maximum sequence cost where each sequence produces distinct gates in PSU(2). The algorithm supports arbitrary universal sets of single-qubit base gates with individually assigned cost values. The database can then be searched to find a sequence approximating a specified target gate. We use this algorithm to compare the cost of cost-optimal gate synthesis between various sets of code level logical base gates consisting of Clifford gates and Z-rotation gates from higher orders of the Clifford hierarchy. The logical base gate costs assigned are total average numbers of raw magic states consumed during the corresponding magic state distillation and implementation procedures. Each set of logical base gates is compared by calculating how the average gate sequence cost for approximating random target gates scales with respect to reaching target gate synthesis logical error rates. We find that by including Z-rotations from the fourth order of the Clifford hierarchy as logical base gates, in addition to the standard Clifford+T gates, the average cost-optimal synthesis costs are reduced by up to 30%. This indicates that a significant amount of resources could be saved by adapting current synthesis algorithms to include higher orders of the Clifford hierarchy and to minimise sequence costs with respect to individual gate costs.

In addition, we develop a model to estimate the proportion of sets of Z-rotation gates from higher orders of the Clifford hierarchy among logical base gates within sequences approximating random target gates, without needing to generate the full database of sequences. The proportions calculated in this manner closely fit results obtained using the sequence generation algorithm to approximate uniformly distributed random target gates. The parameters of the
calculation include the maximum sequence cost and separate logical base gate costs for each order of the Clifford hierarchy, which can be readily modified to specify costs for individual logical base gates. The proportions of these sets of logical base gates with chosen implementation costs can be used to help approximate their effectiveness for the purpose of gate synthesis.

RESULTS

Base gates from the Clifford hierarchy

The Clifford hierarchy is an infinite discrete set of gates that are universal for the purposes of quantum computation and can be fault-tolerantly performed on certain error-correcting codes. Each order of the hierarchy is defined as

$$C_l := \{ U \mid U P U^\dagger \in C_{l-1}, \forall P \in \mathcal{P} \},$$

noting that $$C_1 = \mathcal{P}$$ is the set of Pauli gates, $$C_2$$ is the set of Clifford gates and $$C_3$$ includes, among others, the Pauli basis rotations by $$\pi/4$$ such as the $$T$$ gate. Higher order gates typically correspond to finer angle rotations. Calculating precise resource costs of implementing each gate fault-tolerantly is an extensive task that would need to consider a variety of factors such as qubit count, circuit depth, magic state distillation cost and details of the error-correction implementation. As an approximation for the cost of these code level logical gates, we use the average number of raw magic states consumed to distil and implement them using their corresponding magic state distillation process. Gates from $$C_1$$ and $$C_2$$ are assumed to be free resources since they can be implemented relatively easily. For higher orders of the Clifford hierarchy, resource costs have been calculated for $$Y$$-rotation gates by searching for optimal combinations of various distillation protocols with respect to target gate synthesis error rates $$\epsilon$$ [21]. To follow convention, the $$Y$$-rotation gates are converted to $$Z$$-rotation gates with the same cost using the relation $$R_y(\theta) = H S^\dagger R_y(\theta) S H$$, since $$H$$ and $$S := R_y(\pi/2)$$ have zero cost due to being elements of $$C_2$$. These resource costs vary between orders of the Clifford hierarchy and are shown in Table 3.

In this work, we compare sets of single-qubit universal logical base gates consisting of Clifford gates and $$Z$$-rotation gates from higher orders of the Clifford hierarchy. Although only higher order $$Z$$-rotations are included, they can be readily converted to other gates in the same order of the Clifford hierarchy by multiplying gates from lower orders. In particular, by multiplying Clifford gates, other gates of the same order are generated for the same cost. Higher order $$Z$$-rotations are included, they can be readily converted to other gates in the same order of the Clifford hierarchy by multiplying gates from lower orders. For example $$Z R_z(\pi/4) = R_z(5\pi/4)$$ and $$H R_z(\pi/4) H = R_z(\pi/4)$$ up to global phase, where $$H$$ is the Hadamard gate and $$Z$$ is the Pauli-$$Z$$ gate. These sets of logical base gates are compared with respect to the optimal resource costs resulting from gate synthesis for random target gates. Each set of $$Z$$-rotation gates from order 3 up to global phase, where $$H$$ is the Hadamard gate and $$Z$$ is the Pauli-$$Z$$ gate. These sets of logical base gates are compared with respect to the optimal resource costs resulting from gate synthesis for random target gates. Each set of $$Z$$-rotation gates from order 3 ≤ i ≤ 7 of the Clifford hierarchy, denoted $$\mathcal{T}_i$$, can be written as

$$\mathcal{T}_3 := \left\{ R_z \left( \frac{k \pi}{4} \right) \in C_3 \mid k \in \{-1,1\} \right\},$$

$$\mathcal{T}_4 := \left\{ R_z \left( \frac{k \pi}{8} \right) \in C_4 \mid k \in \{-3,-1,1,3\} \right\},$$

$$\mathcal{T}_5 := \left\{ R_z \left( \frac{k \pi}{16} \right) \in C_5 \mid k \in \{-7,-5,\ldots,5,7\} \right\},$$

$$\mathcal{T}_6 := \left\{ R_z \left( \frac{k \pi}{32} \right) \in C_6 \mid k \in \{-15,-13,\ldots,13,15\} \right\},$$

$$\mathcal{T}_7 := \left\{ R_z \left( \frac{k \pi}{64} \right) \in C_7 \mid k \in \{-31,-29,\ldots,29,31\} \right\}.$$ (2)

The five sets of logical base gates used in our analysis are then constructed as

Set_1 := C_1 \cup C_2 \cup \mathcal{T}_3,
Set_2 := \text{Set}_1 \cup \mathcal{T}_4,
Set_3 := \text{Set}_2 \cup \mathcal{T}_5,
Set_4 := \text{Set}_3 \cup \mathcal{T}_6, \quad \text{and}
Set_5 := \text{Set}_4 \cup \mathcal{T}_7. \quad \text{(3)}
TABLE I. The average raw magic state count required for distillation and implementation of corresponding logical gates, obtained from the supplementary materials of [21]. Each column contains the cost of implementing a logical $Z$-rotation gate from order $i$ of the Clifford hierarchy $T_i$ to below a logical gate error rate $\mu$ calculated using the diamond norm. The raw magic state physical level error is assumed to be 0.1%.

| Error Rate $\mu$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ |
|------------------|-------|-------|-------|-------|-------|
| $10^{-5}$        | 5.1   | 16.7  | 34.8  | 49.0  | 64.7  |
| $10^{-10}$       | 36.2  | 103.1 | 272.7 | 255.8 | 344.8 |
| $10^{-15}$       | 70.4  | 186.5 | 333.2 | 486.1 | 671.5 |
| $10^{-20}$       | 120.1 | 358.7 | 635.8 | 962.2 | 1351.2 |

FIG. 1. An example of a sequence tree used to relate logical base gates, gate sequences and combined gates for the sequence generation algorithm. A node $n$ corresponds to a single-qubit base gate $B_n$ and the root node corresponds to the identity gate $B_0 = I$. A gate sequence corresponding to $n$ is the sequence of logical base gates along the path from $B_0$ to $B_n$. A combined gate $S_n$ is calculated by multiplying all logical base gates within the gate sequence in sequence order. In this example, $B_1$, $B_2$ and $B_3$ are logical base gates where $B_1 = B_4 = B_7 = B_{10}$, $B_2 = B_5 = B_8 = B_{11}$ and $B_3 = B_6 = B_9 = B_{12}$. In the sequence generation algorithm, the leaf node with the lowest sequence cost is expanded by adding a child node as a new leaf node for each gate in the set of logical base gates. All non-leaf nodes of the tree correspond to cost-optimal sequences and they can be thought of as the cost-optimal sequence database generated by the algorithm. Although all leaf nodes are depicted to be at the same depth in the tree, this is not always the case. At any point during the sequence generation algorithm, a path of relatively expensive logical base gates may be much shorter than a path of relatively cheap gates.

Sequence Generation Algorithm

In this section, a sequence generation algorithm, based on Dijkstra’s algorithm, is developed that generates a database of all cost-optimal single-qubit gate sequences below some maximum cost using arbitrary sets of universal base gates which have individually assigned cost values. We use this algorithm to help study the average cost of cost-optimal gate synthesis when including $Z$-rotation gates from higher orders of the Clifford hierarchy as base gates. Due to the flexibility of this algorithm, it could be used as a subroutine within other synthesis algorithms. For example, it could be used as the base approximation step within the SK algorithm, enabling the SK algorithm to consider individual base gate costs when synthesising target gates.

The sequence generation algorithm explores the space of sequence configurations using a tree expansion as shown in Figure 1 where each node corresponds to a gate and each path from the root node to any other node corresponds to a sequence of gates. Let $B_n$ be an element of $PSU(2)$ corresponding to the base gate of node $n$ in the sequence tree. A combined gate $S_n$ of node $n$ is calculated by multiplying all nodes within the branch from the root down to $n$, i.e. $S_n := B_{n_0} \cdot B_{n_1} \ldots B_{n_k}$ where $n_i$ is the $i^{th}$ node from the root node such that $n_0$ is the root and $n_k$ is node $n$. The Lie algebra generator of $S_n$ in the Pauli basis is of the form of a vector $\alpha_n X + \beta_n Y + \gamma_n Z$ with real coefficients and can be written as $(\alpha_n, \beta_n, \gamma_n)$. Each vector represents a point in a ball of radius $\pi/2$ over the Pauli bases $X$, $Y$ and $Z$. Thus each point within the ball is a geometrical location corresponding to a single-qubit gate.
The geometric nearest-neighbour access tree \cite{25} and vantage point tree \cite{23, 24} may be used instead. In general, further alternative data structures may be used such as a sieve and help to avoid duplicate patterns.

In Algorithm 1 cost-optimal sequences and their corresponding vectors are stored in a k-d tree which uses the Euclidean distance on the vectors to organise the data. Due to the periodic nature of the vectors, there is a small chance of failure in the k-d tree when searching for nearest neighbours to points close to the boundary. With computational overhead, the k-d tree may be modified to help overcome this \cite{22}, or a more appropriate data structure such as a vantage point tree \cite{23, 24} may be used instead. In general, further alternative data structures may be used such as the geometric nearest-neighbour access tree \cite{25}.

The pseudocode for the algorithm is shown in Algorithm 1. It works by expanding nodes in a sequence tree (see Figure 1). All leaf (end) nodes of the sequence tree are stored in a minimum heap data structure which sorts the leaf nodes based on their corresponding sequence cost in increasing order. This determines the order of nodes to expand. The tree begins as a single identity gate at the root node which is added as the first element to the leaf node heap. At each iteration, the leaf node with the lowest sequence cost, $i$, is taken from the heap, which for the first iteration would be the identity gate node. The vector $(\alpha_i, \beta_i, \gamma_i)$ is calculated from the combined gate of the corresponding node’s sequence. Before expanding a node in the sequence tree, we check whether another node with the same combined gate vector has already been expanded, using a hashset data structure. If the vector exists in the hashset, then the node is removed from the sequence tree and the algorithm proceeds to the next iteration. This repeats until a unique vector is found. When such a vector is found, it is added to the hashset for uniqueness checking in further iterations and the corresponding node in the sequence tree is expanded by generating a child node for each base gate. Each of these child nodes are added to the leaf node heap. To save computation time, adding a child node to the sequence tree and the heap can be limited to when their corresponding vectors are unique. Since vectors of sequences with lower costs are always added to the hashset before those with higher costs, the hashset must only contain vectors corresponding to sequences with the lowest cost among all sequences that produce equivalent combined gates. Thus, whenever a vector is successfully added to the hashset, the corresponding sequence must be cost-optimal. The cost-optimal vector and sequence pair can be stored in a data structure such as a k-d tree which can be used to approximate target gates by geometrically searching for nearest neighbours in the space of vectors.

There is a notable further optimisation that could be implemented into Algorithm 1. During the procedure, all non-leaf nodes within the sequence tree correspond to cost-optimal sequences with unique vectors. Using the sequence tree itself to guide node expansion, instead of assessing all base gate child nodes for each node expansion, can act as a sieve and help to avoid duplicate patterns.

In Algorithm 1 cost-optimal sequences and their corresponding vectors are stored in a k-d tree which uses the Euclidean distance on the vectors to organise the data. Due to the periodic nature of the vectors, there is a small chance of failure in the k-d tree when searching for nearest neighbours to points close to the boundary. With computational overhead, the k-d tree may be modified to help overcome this \cite{22}, or a more appropriate data structure such as a vantage point tree \cite{23, 24} may be used instead. In general, further alternative data structures may be used such as the geometric nearest-neighbour access tree \cite{25}.

\begin{algorithm}
\begin{algorithmic}[1]
\Procedure{Cost-optimal sequence generation}{baseGates, maxCost}
\State sequenceDatabase $\leftarrow$ new KdTree(Node)
\State sequenceTree $\leftarrow$ new Tree(Node)
\State sequenceTree.SetRoot(Identity gate)
\State sortedLeafNodes $\leftarrow$ new MinHeap(Node)
\State uniqueVectors $\leftarrow$ new Hashset(Vector3)
\State Add sequenceTree.root to sortedLeafNodes
\While{sortedLeafNodes not empty}
\State $i \leftarrow$ sortedLeafNodes.Pop()
\If{sequenceTree.SequenceCost($i$) $>$ maxCost}
\State \Return sequenceDatabase
\EndIf
\State $(\alpha_i, \beta_i, \gamma_i) \leftarrow$ sequenceTree.GetVector($i$)
\If{$(\alpha_i, \beta_i, \gamma_i)$ not in uniqueVectors}
\State Add $i$ to sequenceDatabase
\State Add $(\alpha_i, \beta_i, \gamma_i)$ to uniqueVectors
\EndIf
\ForAll{$j$ in childNodes}
\State $(\alpha_j, \beta_j, \gamma_j) \leftarrow$ sequenceTree.GetVector($j$)
\If{$(\alpha_j, \beta_j, \gamma_j)$ not in uniqueVectors}
\State Add $j$ to sortedLeafNodes
\State Remove $j$ from sequenceTree
\EndIf
\EndFor
\EndWhile
\EndProcedure
\end{algorithmic}
\end{algorithm}
FIG. 2. Cost-optimal sequence costs averaged over 5000 random target gates with respect to target gate synthesis logical error rates $\epsilon$. The logical base gates used are specified in Eq. 3 with cost values (shown in Table I) assigned as the average number of raw magic states required to distil and implement them to below a specified logical gate error. The synthesis logical errors $\epsilon$ are calculated using the trace distance (shown in Equation 4). Corresponding linear best fit values are shown in Table II. The pattern of the data about the lines of best fit for each logical base gate set are similar between plots because for each of the logical base gate errors, the ratios of the base gate cost values between orders of the Clifford hierarchy are similar, hence the cost optimal sequences will be comparable. (a) Synthesis using logical base gate costs associated with $\mu = 10^{-5}$ logical base gate error. (b) Synthesis using logical base gate costs associated with $\mu = 10^{-10}$ logical base gate error. (c) Synthesis using logical base gate costs associated with $\mu = 10^{-15}$ logical base gate error. (d) Synthesis using logical base gate costs associated with $\mu = 10^{-20}$ logical base gate error.

Synthesis Results

Algorithm 1 was used with sets of logical base gates from Eq. 3 and corresponding cost values from Table I to generate a database, which is in the form of a k-d tree, of cost-optimal sequences up to some chosen maximum sequence cost. The sequences were organised in the k-d tree with respect to the vectors corresponding to their combined gates. For a given target gate $G$, gate synthesis was performed by searching for the lowest cost sequence among all nearest neighbours of $G$ up to a chosen synthesis error (distance), $\epsilon$, between their combined gates and $G$. The errors were computed using the trace distance defined as

$$\text{dist}(S, G) = \sqrt{2 - |\text{tr}(S^\dagger G)|}/2,$$

(4)
where $S$ is a combined gate and $G$ is the target gate. If such a sequence did not exist, then the database was further generated to a higher cost and the process was repeated until a sequence was found. Incrementally generating the cost-optimal sequence database in this manner helps avoid over generation.

Gate synthesis was performed on 5000 random target gates sampled from a uniform distribution for a variety of synthesis error rates $\epsilon$ (calculated using Eq. 4 with respect to the sequences’ combined gates), and for each set of logical base gates and corresponding cost values. The average total sequence costs for each case are shown in Figure 2. Each of the four plots correspond to different resource costs of distilling and implementing the logical base gates with code level logical gate errors $\mu = 10^{-5}, 10^{-10}, 10^{-15}$ and $10^{-20}$ calculated using the diamond norm. The corresponding linear best fit values for each set of logical base gates and corresponding cost values are shown in Table II (physical error rate assumed to be 0.1% in all calculations). The pattern of the data about their lines of best fit for each base gate set are similar between plots. This is because for each of the logical base gate errors, the ratios of the logical base gate cost values between orders of the Clifford hierarchy are similar, hence the cost optimal sequences will be comparable.

We can compare the scaling factors of the fits between different sets of logical base gates to estimate changes in average sequence costs as the synthesis error $\epsilon$ approaches zero. For logical base gate errors $\mu = 10^{-5}, 10^{-10}, 10^{-15}$ and $10^{-20}$, we find that Set3 provides 23 ± 3%, 27 ± 3%, 30 ± 3% and 26 ± 3% reductions in scaling factor respectively compared to Set1. For $\mu = 10^{-10}$ and $10^{-15}$, we find that Set4 provides 30 ± 4% and 33 ± 3% reductions in scaling factor respectively compared to Set1, which are both approximately a further 3% reduction to Set2. No further improvements are noticeable in our data.

These results show that for any error-correction scheme with distillation costs assigned according to Table I using Set2 (which includes $T_4$ as logical base gates) instead of the standard Set1, reduces the average resource cost scaling factor with respect to the synthesis negative log-error, $\log(\epsilon^{-1})$, by up to 30%. Additionally Set3 can provide up to a further 3% reduction when compared to Set2. These reductions in resource costs indicate that the resource requirements of synthesis algorithms may be significantly improved by including higher orders of the Clifford hierarchy as logical base gates and by considering individual costs of implementing them.

### Modelling Gate Proportions

The sets of logical base gates Set3, Set4 and Set5 (see Eq. 3) were shown to provide marginal resource savings for gate synthesis when compared with Set2 (see Fig 2), even though they contain many more logical base gates. To investigate this behaviour we develop a model in Appendix I for determining the proportion of sets of gates among all $T_n$ gates where $n \geq 3$ within cost-optimal sequences approximating random target gates with specified gate costs. The proportions can provide insight into how the average sequence cost changes with respect to which $T_n$ base gates are included as logical base gates and what cost values are assigned. For logical base gates with non-zero proportion

---

### Table II

| Base Gates | Scaling Factor | Constant |
|------------|----------------|----------|
| Set1       | 52.4 ± 1.3     | -43.5 ± 2.2 |
| Set2       | 40.6 ± 0.9     | -30.7 ± 1.5 |
| Set3       | 40.8 ± 1.8     | -31.0 ± 2.6 |
| Set4       | 40.8 ± 1.8     | -31.0 ± 2.6 |
| Set5       | 40.8 ± 1.8     | -31.0 ± 2.6 |

(a) Linear fits for Figure 2a for below $\mu = 10^{-5}$ logical base gate error

| Base Gates | Scaling Factor | Constant |
|------------|----------------|----------|
| Set1       | 722 ± 15       | -599 ± 27 |
| Set2       | 503 ± 11       | -370 ± 17 |
| Set3       | 482 ± 10       | -347 ± 16 |
| Set4       | 488 ± 21       | -355 ± 30 |
| Set5       | 488 ± 21       | -355 ± 30 |

(c) Linear fits for Figure 2c for below $\mu = 10^{-15}$ logical base gate error

| Base Gates | Scaling Factor | Constant |
|------------|----------------|----------|
| Set1       | 1230 ± 30      | -1020 ± 50 |
| Set2       | 913 ± 24       | -680 ± 39 |
| Set3       | 893 ± 41       | -661 ± 59 |
| Set4       | 893 ± 41       | -661 ± 59 |
| Set5       | 893 ± 41       | -661 ± 59 |

(d) Linear fits for Figure 2d for below $\mu = 10^{-20}$ logical base gate error
FIG. 3. This figure shows the summed proportions of logical base gates from sequences resulting from the sequence generation algorithm and the proportions calculated using our model. The sequence generation algorithm outputs cost-optimal sequences approximating random target gates to within $\epsilon = 0.03$ synthesis logical gate error under the trace distance (see Eq. (4)), while the model outputs the proportion of a set of logical base gates within the space of all cost-optimal sequences below a maximum cost that produce distinct combined gates. Clifford gates are ignored in the calculations since they are assumed to have zero cost. Both plots show that the model data closely fit the corresponding results from the sequence generation algorithm. The data show that increasing the logical base gate distillation and implementation cost of a particular set $c_T$ drastically lowers the proportion of them found within the generated cost-optimal sequences. Thus the set $c_T$ with increased costs becomes less effective at reducing the average cost-optimal sequence costs, since they are found less frequently within the sequences. Logical base gate costs are assigned according to Table I with a logical base gate error of $\mu = 10^{-15}$ calculated using the diamond norm. The red, green and blue vertical lines (ordered left to right) indicate the logical base gate distillation and implementation costs for $c_T$, $c_T'$ and $c_T''$ respectively. (a) The summed proportions of $c_T$ logical base gates among $c_T \cup c_T'$ gates for cost-optimal sequences consisting of Set 2 logical base gates. Logical base gates from $c_T'$ are fixed while the cost for $c_T$ gates vary. (b) The summed proportions of $c_T'$ logical base gates among $c_T \cup c_T' \cup c_T''$ gates for cost-optimal sequences consisting of Set 3 logical base gates. Logical base gates from $c_T \cup c_T'$ are fixed while the cost for $c_T''$ gates vary.

within sequences approximating target gates, we expect that by increasing their cost, their recalculated proportion will decrease and the average cost of these sequences will increase. Furthermore, for sets of logical base gates with relatively small proportions, the average sequence cost would only slightly increase if the set were to be excluded compared to sets of base gates with larger proportions.

The model estimates the average proportion, $p_n$, of $c_T$ logical base gates among all $c_T$ gates where $i \geq 3$ from within cost-optimal sequences approximating random target gates to within sufficiently small synthesis errors $\epsilon$. The construction is based on a unique canonical form [13] for sequences of logical base gates and is defined as

$$c.t_1.H.t_2.H \ldots t_N.c', \tag{5}$$

where $c$ and $c'$ are Clifford gates, $H$ is the Hadamard gate, $t_j$ is the $j$th positioned gate from order three and above of the Clifford hierarchy, and $N$ is the total number of $t_j$ gates in the sequence. This canonical form has the property that arbitrary gate sequences with distinct combined gates, where the sequences can consist of logical base gates from the Clifford gates and $Z$-rotations from orders three and above of the Clifford hierarchy, can be reduced to distinct sequences of this form. The gate proportion for $c_T$, denoted $p_n$, can be calculated by averaging the $c_T$ logical base gate count over all possible sequences in this canonical form that are below a chosen maximum cost $C$. That is,

$$p_n = \frac{\sum \text{[C/c]}}{\sum \text{[C/c]}} \frac{\text{[(C-c_kk)]/c_k]}{\sum \text{[C-c_kk]/c_k}}} \frac{\text{[(C-c_kk_k)/c_k]}\cdots \text{[(C-c_kk_k_k)/c_k_k_k_k]}}{\sum \text{[C-c_kk_k_k]/c_k_k_k_k}} k_n \left( \prod_{t=3}^{L} k_t \right) \prod_{i=3}^{L} \frac{k_i}{k_i t} , \tag{6}$$

where $c$ and $c'$ are Clifford gates, $H$ is the Hadamard gate, $t_j$ is the $j$th positioned gate from order three and above of the Clifford hierarchy, and $N$ is the total number of $t_j$ gates in the sequence. This canonical form has the property that arbitrary gate sequences with distinct combined gates, where the sequences can consist of logical base gates from the Clifford gates and $Z$-rotations from orders three and above of the Clifford hierarchy, can be reduced to distinct sequences of this form. The gate proportion for $c_T$, denoted $p_n$, can be calculated by averaging the $c_T$ logical base gate count over all possible sequences in this canonical form that are below a chosen maximum cost $C$. That is,
where $c_i$ is the logical base gate distillation and implementation cost for $T_i$, $k_i$ is the number of $T_i$ within a particular sequence, $l_i$ is the number of gates within $T_i$, and $L$ is the order of the Clifford hierarchy to include $Z$-rotation gates up to.

This calculation outputs values closely matching proportion results obtained using the sequence generation algorithm for random target gates, as shown in Figure 3. Figure 3a shows the summed proportions of all $T_4$ gates among $T_3 \cup T_4$ gates over a variety of $T_4$ cost values for sequences consisting of Set2 logical base gates. Figure 3b shows the summed proportions of all $T_5$ gates among $T_3 \cup T_4 \cup T_5$ gates over a variety of $T_5$ cost values for sequences consisting of Set3 logical base gates. The other logical base gate costs are assigned values according to their distillation and implementation cost with a maximum logical error of $\mu = 10^{-15}$ as shown in Table I. These results suggest that increasing the logical base gate distillation and implementation cost of a set $T_n$ drastically lowers the proportion of them found within the database of cost-optimal sequences. Thus they become less effective at reducing the average cost-optimal sequence costs since they are included within sequences less often.

This is a simpler calculation compared to actually performing gate synthesis for many random target gates. So it could be used to help provide a quick indication for the effectiveness of reducing average cost-optimal sequence costs for single-qubit gate synthesis when including a $Z$-rotation logical base gate from the Clifford hierarchy with a specified distillation and implementation cost.

**DISCUSSION**

We investigate the cost of sequences produced by cost-optimal single-qubit gate synthesis using logical base gates from a combination of Clifford gates and $Z$-rotation gates from higher orders of references. An algorithm, based on Dijkstra’s algorithm, was used to generate a database of cost-optimal sequences from arbitrary single-qubit universal sets of logical base gates with individually assigned costs. As logical base gates, combinations of Clifford gates and $Z$-rotation gates from various orders of the Clifford hierarchy were used and assigned costs according to the average number of raw magic states used to distil and implement them in error-correction codes.

Gate synthesis was then performed by finding nearest neighbours within the database of cost-optimal sequences in the Pauli vector space corresponding to combined gates of sequences. We found that by including the fourth order $Z$-rotation gates from the Clifford hierarchy along with the standard Clifford+$T$ gate set, the average cost-optimal sequence costs decreased by up to 30%. We observe up to a further 3% decrease when additionally including the $Z$-rotation gates from the fifth order. No noticeable improvement is observed when additionally including higher order $Z$-rotation logical base gates up to the seventh order.

We investigated this behaviour by developing a model that estimates the proportion of logical base gates within sequences approximating random target gates. This model assumes that each $Z$-rotation gate from orders three and above of the Clifford hierarchy have equal proportions when assigned equal cost values, that is, the gate operations have equal usefulness for approximating random target gates for the purposes of gate synthesis. The proportion estimations were shown to closely fit the data obtained using the sequence generation algorithm on random target gates. This suggests that the lack of observed cost reduction when using higher order logical base gates is due to there being far less numbers of them at their assigned costs within all cost-optimal sequences generated up to the chosen maximum sequence cost. Thus the frequency of the base gates being used for synthesis of random target gates is low, leading to a low level of influence over the average resource costs overall. The model provides a simple method, without needing to generate the full database of sequences, for estimating these gate proportions with each order of the Clifford hierarchy being assigned individual cost values. Thus it could be used to help approximate the effectiveness of sets of logical base gates with chosen implementation costs for the purpose of gate synthesis.

**ACKNOWLEDGEMENTS**

This work was supported by the University of Melbourne through the establishment of an IBM Network Q Hub at the University. CDH is supported by a research grant from the Laby Foundation. We would like to thank Earl Campbell and Kae Nemoto for valuable discussions.

[1] Sergey B Bravyi and A Yu Kitaev. Quantum codes on a lattice with boundary. *arXiv preprint quant-ph/9811052*, 1998.

[2] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002.
[3] Robert Raussendorf, Jim Harrington, and Kovid Goyal. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics*, 9(6):199, 2007.

[4] David S Wang, Austin G Fowler, and Lloyd CL Hollenberg. Surface code quantum computing with error rates over 1%. *Physical Review A*, 83(2):020302, 2011.

[5] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. *Physical Review Letters*, 102(11):110502, 2009.

[6] Xinlan Zhou, Debbie W Leung, and Isaac L Chuang. Methodology for quantum logic gate construction. *Physical Review A*, 62(5):052316, 2000.

[7] Christopher M. Dawson and Michael A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Information & Computation*, 6(1):81–95, 2006.

[8] Austin G Fowler. Constructing arbitrary Steane code single logical qubit fault-tolerant gates. *Quantum Information & Computation*, 11(9-10):867–873, 2011.

[9] A Yu Kitaev, AH Shen, and MN Vyalyi. Classical and Quantum Computation (Graduate Studies in Mathematics vol 47)(Providence, RI: American Mathematical Society). 2002.

[10] Ken Matsumoto and Kazuyuki Amano. Representation of quantum circuits with Clifford and $\pi/8$ gates. *arXiv preprint arXiv:0806.3834*, 2008.

[11] Alex Bocharov and Krysta M Svore. Resource-optimal single-qubit quantum circuits. *Physical Review Letters*, 109(19):190501, 2012.

[12] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single-qubit unitaries generated by Clifford and $T$ gates. *Quantum Information & Computation*, 13(7–8):607–630, 2013.

[13] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Practical approximation of single-qubit unitaries by single-qubit quantum Clifford and $T$ circuits. *IEEE Transactions on Computers*, 65(1):161–172, 2016.

[14] Neil J Ross and Peter Selinger. Optimal ancilla-free Clifford+$T$ approximation of $z$-rotations. *arXiv preprint arXiv:1403.2975*, 2014.

[15] Simon Forest, David Gosset, Vadym Kliuchnikov, and David McKinnon. Exact synthesis of single-qubit unitaries over Clifford-cyclotomic gate sets. *Journal of Mathematical Physics*, 56(8):082201, 2015.

[16] Vadym Kliuchnikov, Alex Bocharov, Martin Roetteler, and Jon Yard. A framework for approximating qubit unitaries. *arXiv preprint arXiv:1510.03888*, 2015.

[17] Peter Selinger. Efficient Clifford+$T$ approximation of single-qubit operators. *Quantum Information & Computation*, 15(1–2):159–180, 2015.

[18] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Asymptotically optimal approximation of single qubit unitaries by Clifford and $T$ circuits using a constant number of ancillary qubits. *Physical Review Letters*, 110(19):190502, 2013.

[19] Alex Bocharov, Martin Roetteler, and Krysta M Svore. Efficient synthesis of probabilistic quantum circuits with fallback. *Physical Review A*, 91(5):052317, 2015.

[20] Daniel Gottesman and Isaac L Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390, 1999.

[21] Earl T Campbell and Joe O’Gorman. An efficient magic state approach to small angle rotations. *Quantum Science and Technology*, 1(1):015007, 2016.

[22] Joshua M Brown, Terry Bosomaier, and Lionel Barnett. Review of data structures for computationally efficient nearest-neighbour entropy estimators for large systems with periodic boundary conditions. *Journal of Computational Science*, 23:109–117, 2017.

[23] Jeffrey K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179, 1991.

[24] Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Soda*, volume 93, pages 311–21, 1993.

[25] Tien Trung Pham, Rodney Van Meter, and Clare Horsman. Optimization of the Solovay-Kitaev algorithm. *Physical Review A*, 87(5):052332, 2013.
Appendix A: Model for Gate Proportions

Here we develop the theory for estimating the average proportion \( p_n \) where \( n \geq 3 \) of logical base gates from \( \mathcal{T}_n \) found within cost-optimal sequences approximating random target gates within a synthesis error threshold of \( \epsilon \), ignoring Clifford gates. We begin by assuming that each logical base gate in \( \mathcal{T}_3 \cup \mathcal{T}_4 \ldots \mathcal{T}_n \) for \( n \geq 3 \) has equal proportions if they were to have equal costs, that is, the gate operations are equally effective for the purposes of gate synthesis. This can be justified by the data in Figure A.1. It shows that when each logical base gate is given equal costs, the sequence generation algorithm generates a database of gate sequences with each gate having approximately the same proportions, where the proportions slowly decrease for increasing order. We do not expect these proportions to significantly change for larger sequence costs (or smaller synthesis error thresholds \( \epsilon \)), since the logical base gate proportions are approximately constant for sufficiently large maximum sequence costs. This can be seen in Fig. A.2, for the case of \( \mathcal{T}_5 \) logical base gates from within Set 3 generated by the sequence generation algorithm for random target gates.

Assume we have a database of cost-optimal gate sequences that have been generated up to a chosen maximum cost with individually assigned distillation and implementation costs for each set of logical base gates \( \mathcal{T}_n \) where \( n \geq 3 \). We will calculate the proportion of \( \mathcal{T}_n \) gates among all sequences within the database.

For simplicity, let logical gates from \( \mathcal{T}_n \) be called \( t \) gates. Using a unique canonical form \([15]\) for sequences consisting of the Clifford gates and combinations of \( \mathcal{T}_n \), arbitrary gate sequences can be reduced to the form

\[
c.t_1,H,t_2,H \ldots t_N,c',
\]

where \( c \) and \( c' \) are Clifford gates, \( t_j \) is the \( j \)th positioned \( t \) gate in the sequence, and \( N \) is the \( t \)-count. For a particular sequence, let the number of \( t \) gates from \( \mathcal{T}_i \) be denoted by \( k_i \). It follows that each sequence consisting of gates from up to order \( L \) of the Clifford hierarchy satisfies (noting that \( c_0 = c_1 = 0 \))

\[
\sum_{i=3}^{L} c_i k_i \leq C,
\]

where \( c_i \) is the cost assigned to logical gates from \( \mathcal{T}_i \) and \( C \) is the maximum cost of the database of gate sequences. It will be useful to denote the number of \( t \) gates from order \( n \) and above of the Clifford hierarchy as

\[
K_n := \sum_{i=n}^{L} k_i,
\]

noting that \( K_3 = N \) is the \( t \)-count.

To calculate the proportions among all sequences within the database, we first count the total number of possible sequences that can be formed given a set of \( t \) gate counts \( \{ k_i \}_{i=0}^{L} \) for each sequence. By iterating through every combination of \( t \) gate counts, the total number of \( t \) gates for each order within all sequences satisfying Eq. (A2) can then be summed. For sequences of \( t \)-count \( N \), the number of permutations of \( k_i \) gates within \( N \) gate locations is

\[
(\#\text{Permutations}(k_i,N)) := \binom{N}{k_i} = \frac{N!}{(N-k_i)!k_i!}.
\]

Let \( l_i \) be the number of distinct \( Z \)-rotation gates within order \( i \) of the Clifford hierarchy, for example, \( l_3 = 2 \) since \( \mathcal{T}_3 = \{ T, T^\dagger \} \). Then for each permutation, there are \( l_i^{k_i} \) combinations of assigned \( \mathcal{T}_i \) logical base gates within the permutation. Thus, the total number of configurations for \( k_i \) number of gates with \( l_i \) variations in a sequence of \( t \)-count \( N \) is

\[
\gamma(k_i,l_i,N) := l_i^{k_i} \frac{N!}{(N-k_i)!k_i!}.
\]

After assigning gates to \( k_i \) locations, there are \( N - k_i \) locations remaining within the sequence. When the number of locations remaining is \( K_i \), then we can write \( K_i - k_i = K_{i+1} \). So for each configuration of \( k_i \) gates in \( K_i \) locations, the number of configurations of \( k_{i+1} \) gates with \( l_{i+1} \) variations in the remainder of the sequence of \( K_{i+1} \) gate locations is \( \gamma(k_{i+1},l_{i+1},K_{i+1}) \), giving a total of \( \gamma(k_i,l_i,K_i)\gamma(k_{i+1},l_{i+1},K_{i+1}) \) configurations for \( k_i \) and \( k_{i+1} \) gates in sequences
FIG. A.1. The proportions of individual logical base gates with equally assigned costs (synthesis logical level error 0.03 using the trace distance). The number of gates within each set doubles for increasing order where \( T_3 \) contains two gates (see Equation 2). This plot indicates that the logical base gates are almost equivalently useful in approximating random target gates using cost-optimal gate synthesis.

FIG. A.2. The proportion of \( T_5 \) logical base gates among \( T_3 \cup T_4 \cup T_5 \) gates calculated using the combinatorial model for all cost-optimal sequences below a maximum sequence cost that produce distinct combined gates. The logical base gate cost values are assigned according to Table I for a logical base gate error threshold of \( \mu = 10^{-15} \) under the diamond norm. This plot shows that the proportion of \( T_5 \) gates becomes approximately constant for sufficiently large maximum sequence costs.

of \( t \)-count \( K_i \). Thus the total number of configurations for \( t \) gate counts \( k = (k_3, k_4, \ldots, k_L) \) in sequences of \( t \)-count
N = K_3 (containing t gates up to order L of the Clifford hierarchy) is
\[
\Gamma(k) := \prod_{i=3}^{L} \gamma(k_i, l_i, K_i) = \prod_{i=3}^{L} l_i^{k_i} \frac{K_i!}{(K_i - k_i)! k_i!} \quad (A6)
\]
\[
= \frac{K_3! K_4! \ldots K_L!}{K_4! \ldots K_{L+1}!} \prod_{i=3}^{L} l_i^{k_i} = K_3! \prod_{i=3}^{L} l_i^{k_i} \quad (A7)
\]
\[
= \left( \sum_{i=3}^{L} k_i \right)! \prod_{i=3}^{L} l_i^{k_i} \quad (A8)
\]

To count the total number of sequences, we sum over all configurations for each assignment of \( k \) satisfying Equation [A2]. We begin by determining the maximum allowable values for each \( k_i \) with respect to the lower order \( t \) gate counts \( \{k_j\}_{3}^{t-1} \). The maximum value for \( k_3 \) is \( [C/c_3] \). Given \( k_3 \), the maximum value for \( k_4 \) is \( [(C - c_3 k_3)/c_4] \). By continuing this pattern, given a set of \( t \) gate counts \( \{k_3, k_4, \ldots, k_{t-1}\} \), the maximum value for \( k_i \)
\[
\max(k_i) = [(C - \sum_{j=3}^{i-1} c_j k_j)/c_i]. \quad (A9)
\]

So now the total number of sequence configurations with logical base gate costs \( c \) and maximum sequence cost \( C \) can be calculated as
\[
\zeta(c, C) := \sum_{\{k | c \cdot k \leq C\}} \Gamma(k)
\]
\[
= \sum_{k_3=0}^{[C/c_3]} \sum_{k_4=0}^{[C-c_3 k_3]/c_4} \ldots \sum_{k_L=0}^{[C-c_3 \ldots c_{L-1} k_{L-1}]/c_L} \left( \sum_{i=3}^{L} k_i \right)! \prod_{i=3}^{L} l_i^{k_i} \quad (A10)
\]

Since the number of \( T_n \) logical gates within a particular sequence is \( k_n \), the total proportion of \( T_n \) logical gates among \( t \) gates within all possible sequences below the maximum cost \( C \) is
\[
p_n = \frac{\sum_{\{k | c \cdot k \leq C\}} k_n \Gamma(k)}{\sum_{\{k | c \cdot k \leq C\}} \sum_{i=3}^{L} k_i \Gamma(k)}
\]
\[
= \frac{\sum_{k_3=0}^{[C/c_3]} \sum_{k_4=0}^{[C-c_3 k_3]/c_4} \ldots \sum_{k_L=0}^{[C-c_3 \ldots c_{L-1} k_{L-1}]/c_L} k_n \left( \sum_{i=3}^{L} k_i \right)! \prod_{i=3}^{L} l_i^{k_i}}{\sum_{k_3=0}^{[C/c_3]} \sum_{k_4=0}^{[C-c_3 k_3]/c_4} \ldots \sum_{k_L=0}^{[C-c_3 \ldots c_{L-1} k_{L-1}]/c_L}}. \quad (A12)
\]