Spacetime-Efficient Low-Depth Quantum State Preparation with Applications

Kaiwen Gui[*,1,2,3], Alexander M. Dalzell[*,4], Alessandro Achille[5], Martin Suchara[1], and Frederic T. Chong[3]

[1]Amazon Web Services, WA, USA
[2]Pritzker School of Molecular Engineering, University of Chicago, IL, USA
[3]Department of Computer Science, University of Chicago, IL, USA
[4]AWS Center for Quantum Computing, Pasadena, CA, USA
[5]AWS AI Labs, Pasadena, CA, USA

We propose a novel deterministic method for preparing arbitrary quantum states. When our protocol is compiled into CNOT and arbitrary single-qubit gates, it prepares an N-dimensional state in depth $O(\log(N))$ and *spacetime allocation* (a metric that accounts for the fact that oftentimes some ancilla qubits need not be active for the entire circuit) $O(N)$, which are both optimal. When compiled into the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set, we show that it requires asymptotically fewer quantum resources than previous methods. Specifically, it prepares an arbitrary state up to error ϵ with optimal depth of $O(\log(N) + \log(1/\epsilon))$ and spacetime allocation $O(N \log(\log(N)/\epsilon))$, improving over $O(\log(N) \log(\log(N)/\epsilon))$ and $O(N \log(N/\epsilon))$, respectively. We illustrate how the reduced spacetime allocation of our protocol enables rapid preparation of many disjoint states with only constant-factor ancilla overhead—$O(N)$ ancilla qubits are reused efficiently to prepare a product state of w N-dimensional states in depth $O(w + \log(N))$ rather than $O(w \log(N))$, achieving effectively constant depth per state. We highlight several applications where this ability would be useful, including quantum machine learning, Hamiltonian simulation, and solving linear systems of equations. We provide quantum circuit descriptions of our protocol, detailed pseudocode, and gate-level implementation examples using Braket.

1 Introduction

Quantum state preparation (QSP) is a crucial subroutine in many proposed quantum algorithms that claim speedup over their classical counterparts in applications such as quantum machine learning [1–8], simulating quantum systems [9–12], solving linear systems of equations [13–15], and synthesizing unitary operations [16–18].

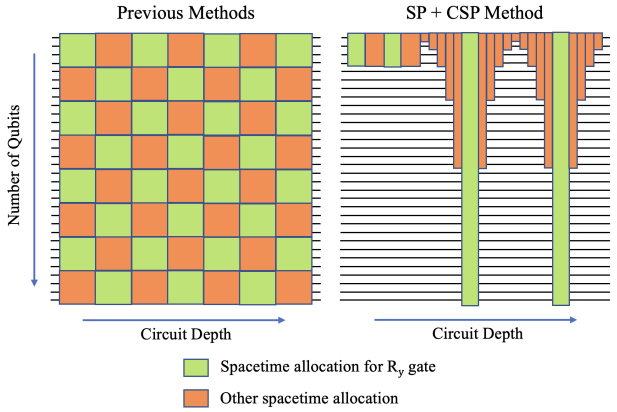

Figure 1: Illustration of spacetime allocation in our state preparation method (SP+CSP) vs. previous methods. Green regions correspond to spacetime allocated for arbitrary single-qubit rotations. Orange regions correspond to spacetime allocated to qubits that are active (i.e., not in $|0\rangle$) without single-qubit rotations; they might experience, e.g., CNOT or Toffoli gates or might be idling while entangled with other qubits. In our method, the spacetime allocation is asymptotically smaller than the product of qubit count and depth.

The state preparation problem is to create an arbitrary quantum state of the form:

$$|\psi\rangle = \frac{1}{\|\mathbf{x}\|} \sum_{i=0}^{2^n-1} x_i \, |i\rangle \tag{1}$$

where n denotes the number of qubits, $\|\cdot\|$ denotes the standard Euclidean norm, and $\mathbf{x}$ is a 2^n-dimensional vector with components $x_i \in \mathbb{C}$. In many applications, it is sufficient to let $x_i \in \mathbb{R}^+ \cup \{0\}$, and henceforth, we assume this for simplicity; the additional phase information can be easily incorporated with minimal overhead (see App. C). We also denote $N = 2^n$ as the total number of parameters encoded, the same as the dimension of the vector space.

*These two authors contributed equally; kgui@uchicago.edu, dalzel@amazon.com

Gate Set	{U(2), CNOT}		{H, S, T, CNOT}	
	Depth	Spacetime Allocation	Depth	Spacetime Allocation
Sun et al. [17]	$O(\log(N)\log(\log(N)))$	$\Theta(N)$	$O(\log(N)\log(\log(N))\log(N/\epsilon))$	$O(N\log(N/\epsilon))$
Sun et al. [17]	$\Theta(\log(N))$	$O(N\log(N))$	$O(\log(N)\log(N/\epsilon))$	$O(N\log(N)\log(N/\epsilon))$
Zhang et al. [19]	$\Theta(\log(N))$	$O(N\log(N))$	$O(\log(N)\log(\log(N)/\epsilon))$	$O(N\log(N)\log(\log(N)/\epsilon))$
Yuan et al. [18]	$\Theta(\log(N))$	$\Theta(N)$	$\Omega(\log(N)\log(\log(N)/\epsilon))^\dagger$	$\Omega(N\log(\log(N)/\epsilon))^\dagger$
Clader et al. [20]	$O(\log^2(N))$	$O(N\log^2(N))$	$O(\log^2(N)+\log(1/\epsilon))$	$O(N(\log^2(N)+\log(1/\epsilon)))$
This Work (SP)	$\boldsymbol{\Theta(\log(N))}$	$\boldsymbol{O(N\log(N))}$	$\boldsymbol{\Theta(\log(N)+\log(1/\epsilon))}$	$\boldsymbol{O(N\log(N/\epsilon))}$
This Work (SP+CSP)	$\boldsymbol{\Theta(\log(N))}$	$\boldsymbol{\Theta(N)}$	$\boldsymbol{\Theta(\log(N)+\log(1/\epsilon))}$	$\boldsymbol{O(N\log(\log(N)/\epsilon))}$

Table 1: Comparison of our results to previous state-of-the-art low-depth QSP methods. $N = 2^n$ is the total number of basis/parameters, and ϵ is the error precision parameter. Note that Sun et al. [17] proposed additional variations of the QSP protocol with larger circuit depth and fewer ancilla qubits that all have spacetime allocation of $\Theta(N)$ and $O(N\log(N/\epsilon))$. The dagger † in the row for Ref. [18] indicates lower bounds from our calculation because the paper did not analyze the Clifford + T costs. Note that T depth is optimized in Clader et al. [20]. We also note that our SP+CSP protocol allows some ancilla registers to be dirty, similar to that in [16]. See the corresponding theoretical lower bound for spacetime allocation using {H, S, T, CNOT} gate set in Sec. 2.3.

Recent advancement by Sun et al. [17] gave an optimal construction that creates $|\psi\rangle$ with $\Theta(2^n/n)$[1] circuit depth using arbitrary single-qubit gates and two-qubit CNOT gates (henceforth called the $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set) and no ancilla qubits. On the other hand, if ancilla qubits are available, one can dramatically reduce the circuit depth. It is preferable to use low-depth protocols when the state-preparation procedure must be completed quickly or repeated many times sequentially. Oftentimes, the quantum algorithm that follows the preparation of $|\psi\rangle$—for instance, making a machine learning inference—runs in depth poly(n), exponentially faster than the ancilla-free QSP implementations, so low-depth state preparation is vital to make the overall runtime reasonable.

Toward that end, recent deterministic methods [17, 19, 21] have achieved optimal $\Theta(n)$ quantum circuit depth in the $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set. However, this exponential depth reduction comes with an exponential overhead in space. In other words, one would need an exponential number of ancilla qubits. Recent work [18] is able to achieve $\Theta(n)$ depth with only $\Theta(2^n/n)$ ancilla qubits, which is optimal.

When considering the total required quantum resource tradeoffs, metrics such as circuit depth, circuit size, and qubit count are typically considered. Assuming the physical architecture allows one to perform gates in parallel, the circuit depth is a proxy for the overall runtime of the computation, and the qubit count represents the overall amount of space that must be allocated to the computation. We now propose another metric—**spacetime allocation**—the total time that each individual qubit must be active (i.e., not in the $|0\rangle$ state), summed over all qubits. The spacetime allocation is bounded below by the circuit size and bounded above by the product of the qubit count and the circuit depth, but it carries a distinct operational meaning. In a model where one wishes to perform many distinct ancilla-intensive jobs (such as rapidly preparing many independent n-qubit states) with a fixed number of ancilla qubits, the availability of fresh ancillae in the state $|0\rangle$ becomes the algorithmic bottleneck. Assuming ancillae can be reallocated from one job to another as soon as they are returned to the $|0\rangle$ state, the overall runtime to complete the batch of jobs is determined by the spacetime allocation of the jobs rather than their depth or size. State-preparation algorithms that are optimal in terms of depth or size are not necessarily also optimal in terms of spacetime allocation. We will show that when compiled into the $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set, our state preparation protocol is simultaneously optimal in depth, size, and spacetime allocation up to constant factors. This feat has already been achieved by the state preparation method of Ref. [18], which has $\Theta(n)$ depth and $\Theta(2^n/n)$ qubit count, along with $\Theta(2^n)$ size and spacetime allocation.

However, in practice, it may not be possible to perform arbitrary single-qubit gates to exact precision, and thus the $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set may not be applicable. In this case, we cannot hope to prepare the state $|\psi\rangle$ exactly; rather, given an error parameter ϵ, we seek to prepare a state $|\tilde{\psi}\rangle$ such that

$$\left\| |\psi\rangle - |\tilde{\psi}\rangle \right\| \leq \epsilon\,. \tag{2}$$

For example, this is the case in most proposals for fault-tolerant quantum computation based on error-correcting codes, where logical single-qubit gates are approximately performed using a sequence of logical gates drawn from a discrete gate set (see, e.g., [22]). A common choice of discrete gate set is $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$

[1] Throughout the paper, we use $O(f(n, 1/\epsilon))$ notation to indicate that the cost is upper bounded by a constant number times $f(n, 1/\epsilon)$ as $n, 1/\epsilon \to \infty$, $\Omega(f(n, 1/\epsilon))$ to denote lower bounds, and $\Theta(f(n, 1/\epsilon))$ when there are upper and lower bounds that match up to constant factors.

(defined in the next section); we quote the scaling of the resource cost of our protocol for approximate QSP when compiled into this gate set. We use the terminology **approximate spacetime allocation** and **exact spacetime allocation**, denoted as $\mathrm{SA}_{\mathrm{approx}}$ and $\mathrm{SA}_{\mathrm{exact}}$, to differentiate the cost in the two models. Similarly, we denote the circuit depth in the two models by $\mathrm{D}_{\mathrm{approx}}$ and $\mathrm{D}_{\mathrm{exact}}$.

Crucially, in the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set, for constant target error ϵ, the method of Ref. [18] does not achieve $\Theta(n)$ depth. While they do not explicitly report the depth in this gate set, this can be seen by the fact that their circuit has $O(2^n)$ arbitrary single-qubit gates spread out over $O(n)$ distinct layers; to convert to the discrete gate set while incurring constant overall error, each single-qubit gate must be approximately decomposed into a sequence of depth at least $\Omega(\log(n))$, yielding total depth at least $\Omega(n \log(n))$. Our state preparation method will have depth $\Theta(n)$ even in the discrete gate set, when ϵ is taken as a constant.

The value of minimizing with respect to spacetime allocation is also apparent in the context of realistic hardware considerations. For example, in gate-based near-term devices without error correction, reducing the spacetime allocation (even as circuit size remains the same) amounts to reducing the time that qubits are left idling. By deallocating and reinitializing ancillas in $|0\rangle$, this kind of idling time can be minimized, and the impacts of noise can be made minimal. Moreover, idling of logical qubits is also costly in fault-tolerant architectures, which require continuous error correction even when no gates are being performed. Each round of error correction requires a set of parity check circuits and measurements, followed by a nontrivial classical decoding calculation, all of which contribute to the overall energy expenditure of the computation.

Another feature of our constructions is that they are garbage-free. This contrasts with some previous work (e.g., [23, 24]), which perform a relaxed version of the QSP task, where, instead of preparing the n-qubit state $|\psi\rangle$ from Eq. (1), one aims to prepare an $(n+a)$-qubit state $|\Psi\rangle$ given by

$$|\Psi\rangle = \frac{1}{\|\mathbf{x}\|} \sum_{i=0}^{2^n-1} \left(x_i \, |i\rangle \otimes |\mathrm{garbage}_i\rangle\right). \tag{3}$$

Typically, a would scale at least linearly in n; these a qubits are left entangled with the n data qubits.

Allowing garbage makes the QSP task easier, and in some applications, garbage is tolerable (since the prepared state only acts as control qubits). However, as long as the a ancilla qubits are entangled with the data, they cannot be used as fresh $|0\rangle$ ancillae for other tasks, and they continue to contribute toward the spacetime allocation of the QSP protocol. Moreover, it is important to emphasize that this is a fundamentally different task. After tracing out the garbage, the n-qubit state that results is a mixed state, which lacks the coherence of the pure state in Eq. (1). This is especially true when one wants to directly manipulate the prepared states (e.g., in quantum linear system solvers and quantum machine learning applications) rather than only using them as control bits.

Lastly, we point out an interesting feature of our state preparation protocol: a constant fraction of the $O(N)$ ancilla qubits needed to achieve logarithmic depth can be *dirty*, that is, they can start in an arbitrary state, and they will be returned to the state they began in at the end of the circuit. Dirty qubits are appealing because they can be prepared without spending resources to fault tolerantly prepare high-fidelity $|0\rangle$ states. Additionally, dirty qubits can be qubits from another computation experiencing a long period of idling, provided they are returned to their original state before the end of the idling period. This situation is especially salient when using the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set, since our state preparation circuits involve a layer where some qubits are idling and others experience a single-qubit gate sequence approximating a single-qubit rotation—the idling qubits can be used as dirty ancillae for other state preparation instances. The possibility of state preparation ancillae being dirty has been previously explored in Ref. [16], which studied tradeoffs between the number of T gates and the number of dirty ancillae. We believe with additional innovations, it may be possible for the number of clean qubits to be further reduced to be asymptotically better than $O(N)$.

The contributions of this work are the following:

1. We propose a novel, deterministic, garbage-free quantum state preparation method, which we call SP+CSP (Section 3), and we show that in the $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set it simultaneously achieves optimal depth and spacetime allocation. We also show that in the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set, it achieves depth and spacetime allocation that are both asymptotically superior to previous methods (see Table 1). A matching lower bound shows that the depth-scaling is optimal in this gate set, whereas the best lower bound on spacetime allocation leaves open the possibility of improvements by logarithmic factors.

2. We show how the optimal spacetime allocation of our protocol allows it to prepare multiple copies of a state more rapidly than other methods (Section 4).

3. We discuss several applications of the optimal spacetime allocation quantum state preparation protocol (Section 5), including

   (a) Quantum Machine Learning
   (b) Hamiltonian simulation
   (c) Solving linear systems with HHL-style quantum algorithms

4. We provide a circuit-level implementation for the SP+CSP method (Sec. 3 & App. A) with detailed pseudocode.

5. We provide two gate-level implementation examples using Braket (Sec. 6).

We summarize our result compared to other state-of-the-art results in Table 1, with a pictorial depiction in Fig. 1.

To achieve these results, we build from the work of Clader et al. [20], who gave a QSP method with depth $O(n)$ and qubit count $O(2^n)$ under the assumption that the FANOUT-CNOT operation—that is, the product of up to $O(2^n)$ CNOT gates sharing the same control qubit—could be performed in a single time step. Our protocol (we call it the **SP** protocol) eliminates the need for FANOUT-CNOT at the expense of only constant-factor overhead in the number of ancilla qubits and the depth. The high-level idea is to use a tree-like data copying circuit consisting only of CNOT gates to copy the control bit into many ancillas, so it can then be used to control many operations in parallel (see App. A.1). However, this observation is not alone sufficient—to achieve $O(n)$ overall depth, we require a delicate method that alternates between layers of control-bit copying and layers of controlled operations (see App. A.4). Like that of Clader et al., this circuit has the feature that only a constant number of layers involve single-qubit rotations (which incur depth $O(\log(n/\epsilon))$ when approximately decomposed into a finite gate set), allowing the overall depth to scale as $O(n + \log(1/\epsilon))$. In comparison, previous methods (e.g., [17, 19]) assemble these single-qubit rotations across at least n layers, which would then make the total depth at least $\Omega(n \log(n/\epsilon))$.

The SP protocol (and protocols designed in previous works such as [17] and [19]) does not achieve optimal spacetime allocation since it fundamentally requires $O(2^n)$ ancilla qubits to be entangled with the $O(n)$ data qubits for a constant fraction of the $O(n)$ circuit depth, leading to $O(n2^n)$ spacetime allocation (similar to what is illustrated on the left side of Fig. 1). To circumvent this, we pursue an additional idea: prepare roughly half of the qubits using the SP method, and then perform *controlled state preparation* (**CSP**) from those qubits into the rest of the qubits. Both SP and CSP require only $O(1)$ layers of single-qubit rotations, preserving the $O(n + \log(1/\epsilon))$ depth for the approximate compilation. Moreover, the full $O(2^n)$ ancilla qubits are only needed very briefly during the CSP procedure, and most can be freed up after only $O(1)$ depth (illustrated as the right side of Fig. 1). Ultimately, we show that $O(2^n)$ spacetime allocation can be achieved. We also give a detailed circuit implementation that shows the difference in spacetime allocation requirements in App. A.7.

2 Background

This section will introduce the required quantum gates and provide more details about the spacetime allocation metric, as well as relevant lower bounds and a summary of the relevant prior work.

2.1 Quantum Gates

In this work, we will use the following quantum gates: X gate, S gate, T gate, Hadamard (H) gate, R_y gate, CNOT gate, SWAP gate, controlled-R_y gate, Fredkin gate, and Toffoli gate.

Gate	Unitary
X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Hadamard (H)	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$
Y-Rotation ($R_y(\theta)$)	$\begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
CNOT	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
T	$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$

Table 2: Elementary Quantum Gates. Note that there is some redundancy as $S = T^2$ and $X = HS^2H$. The R_y gate can be approximated to arbitrary precision as a sequence of H, S, and T gates (see, e.g., [22])

Table 2 shows the elementary quantum gates required for the $\{U(2), CNOT\}$ gate set and the $\{H, S, T, CNOT\}$ gate set. We also summarize all other required quantum gates that can be constructed in Fig. 2, 3, 4, and 5.

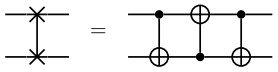

Figure 2: SWAP Gate Decomposition Using CNOT

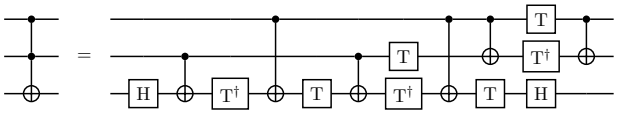

Figure 3: Toffoli (CCNOT) Gate Decomposition Using H, T, CNOT [25]

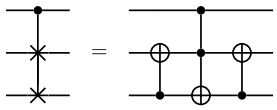

Figure 4: Fredkin (CSWAP) Gate Decomposition Using CNOT, Toffoli [26]

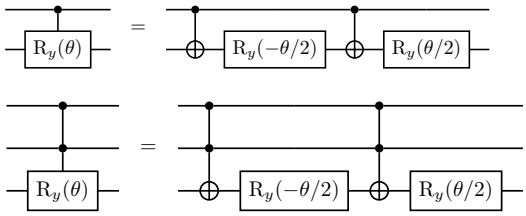

Figure 5: Controlled Rotation Gates (CR_y and CCR_y) Decomposition Using R_y, CNOT, Toffoli

2.2 Spacetime Allocation — QPU "Core Hours"

In classical computing, "core hours" [27] refers to the number of CPUs used to run a certain computing task multiplied by the duration of the job in hours. We now define the a quantum analogue of the classical "core hours" as the quantum spacetime allocation cost, equivalently defined in either of the following ways

- Sum of the individual duration (depth) that each logical qubit is active (i.e., not in the $|0\rangle$ state)
- Sum of the number of active qubits in each layer

Or quantitatively:

$$\mathrm{SA} := \sum_{i=0}^{q-1} d_i = \sum_{t=0}^{d-1} q_t \tag{4}$$

where d is the total depth, q is the total number of qubits, d_i is the active time (depth) for the i^{th} qubit, and q_t is the number of active qubits at layer t.

In the case that all qubits are active for the entirety of the computation, the spacetime allocation is simply the product of the circuit depth and the total number of qubits. However, if most of the ancilla qubits are needed only briefly and can be reset to the $|0\rangle$ state before the end of the computation, the spacetime allocation can be significantly less [28]. In this work, we will show that freeing up the ancilla qubits at early times can bring **asymptotic** spacetime allocation advantage for quantum state preparation.

2.3 Lower Bounds

Previous work has shown QSP lower bounds for circuit depth, size, number of qubits, and the corresponding tradeoffs therein [17, 29] when using only arbitrary single-qubit gates and CNOT gates. Since each gate in the circuit acts on at least one qubit, at least one qubit must be active per gate in each layer. Thus, the number of gates provides a lower bound for the spacetime allocation.

Lemma 2.1. *If a quantum circuit has circuit size* C*, its spacetime allocation must be at least* C*.*

Consequently, if we wish to lower bound the spacetime allocation, it suffices to bound the circuit size. An arbitrary quantum state is described by $O(2^n)$ real parameters. Each single-qubit gate can introduce at most $O(1)$ real parameters, so there must be $\Omega(2^n)$ single-qubit gates. Additionally, any consecutive single-qubit gates on the same qubit that do not have a CNOT in between can be combined into a single gate, so the number of CNOTs must be in the same order as the number of single-qubit gates. This line of reasoning gives rise to a lower bound on circuit size [29].

Lemma 2.2 (Section II of [29])**.** *A quantum circuit consisting of gates drawn from* $\{\mathrm{U}(2), \mathrm{CNOT}\}$ *that prepares an arbitrary* n*-qubit quantum state must have at least* $\Omega(2^n)$ CNOT *gates and at least* $\Omega(2^n)$ U(2) *gates.*

Corollary 2.2.1. *A quantum circuit consisting of gates drawn from* $\{\mathrm{U}(2), \mathrm{CNOT}\}$ *that prepares an arbitrary* n*-qubit quantum state must have at least* $\Omega(2^n)$ *spacetime allocation.*

Similar lower bounds are possible in the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set.

Lemma 2.3. *A quantum circuit consisting of gates drawn from* $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ *that prepares an arbitrary* n*-qubit state to error* ϵ *must have at least* $\Omega\left(\frac{2^n \log(1/\epsilon)}{n+\log(\log(1/\epsilon))}\right)$ *size.*

Proof. We follow the logic similar to Theorem 9 of [30]. Following Lemma 1 in [30], the circuit size C must satisfy $C\log(C) \geq \Omega(2^n \log(1/\epsilon))$. Letting $U = C\log(C)$ and taking the logarithm on both sides, we have $\log(U) = \log(C\log(C)) \geq \log(C)$, and hence $C = U/\log(C) \geq U/\log(U)$. Thus, $U \geq \Omega(2^n \log(1/\epsilon))$ implies $C \geq \Omega(2^n \log(1/\epsilon)/(n + \log(\log(1/\epsilon))))$. □

Note that Ref. [16] has shown that the number of T gates can be significantly smaller than this lower bound, an interesting fact since T gates are significantly more expensive than the other gates in many approaches to fault-tolerant quantum computation.

Corollary 2.3.1. *A quantum circuit consisting of gates drawn from* $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ *for preparing an arbitrary* n*-qubit state to error* ϵ *must have at least* $\Omega\left(\frac{2^n \log(1/\epsilon)}{n+\log(\log(1/\epsilon))}\right)$ *spacetime allocation.*

It is unknown whether the lower bound in Lemma 2.3 and Corollary 2.3.1 is tight.

Lower bounds on depth can also be shown in both gate sets.

Lemma 2.4 (Theorem 3 of [17])**.** *A quantum circuit consisting of gates drawn from* $\{U(2), \mathrm{CNOT}\}$ *that prepares an arbitrary* n*-qubit state using* q *ancilla qubits must have depth at least* $\Omega(\max(n, \frac{2^n}{q+n}))$.

Lemma 2.5. *A quantum circuit consisting of gates drawn from* $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ *that prepares an arbitrary* n*-qubit state using* $O(2^n)$ *ancilla qubits must have depth at least* $\Omega(n + \log(1/\epsilon))$.

Proof. First we examine the n dependence for any constant $\epsilon < 1$. Following a similar argument as [17, Theorem 3], at depth D, the number of gates that are in the "lightcone" of the n data qubits is upper bounded by $n2^D$. The rest of the gates can be ignored. Since there are a constant number of gates in the gate set, the total number of unique output states is upper bounded by $e^{O(n2^D)}$. Meanwhile, Eq. (4.85) of Ref. [31] asserts that the number of circuits needed to cover the entire set of n-qubit states up to $\epsilon = O(1)$ precision is at least $e^{\Omega(2^n)}$. Together, these imply that $D = \Omega(n)$. Separately, we can lower bound the ϵ dependence as $D = \Omega(\log(1/\epsilon))$ directly from Theorem 9 of [30], once we set $n_{anc} = O(2^n)$. These two bounds hold independently for sufficiently large n and sufficiently small ϵ, thus the sum of the bounds must also hold (up to a constant factor that can be absorbed into big-Ω), implying the overall stated bound of $\Omega(n + \log(1/\epsilon))$. □

The SP+CSP circuit construction illustrated in Sec. 3 will give an upper bound that matches the lower bound of Lemma 2.5.

2.4 Upper Bounds in Prior Work

2.4.1 $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set

To the best of our knowledge, the state-of-the-art quantum state preparation [17, 18] methods achieve the following depth, where a is the number of ancillae available:

$$\mathrm{D}_{\mathrm{exact}} = \begin{cases} \Theta(\frac{2^n}{n}), & \text{if } a = 0 \text{ (need } n \text{ data qubits)} \\ \Theta(\frac{2^n}{n+a}), & \text{if } a = O(2^n/n) \\ \Theta(n), & \text{if } a = \Omega(\frac{2^n}{n}) \end{cases}$$

Here $\mathrm{D}_{\mathrm{exact}}$ labels the depth of the exact quantum state preparation circuit using the $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set, a denotes the number of ancillae, and n denotes the number of qubits of the desired arbitrary quantum state to be prepared.

We can compute the spacetime allocation upper bound by simply multiplying the depth with the number of qubits required; we find that the lower bound of $\Omega(2^n)$ is saturated for any choice of ancilla qubits $a < O(2^n/n)$, and in particular:

$$\mathrm{SA}_{\mathrm{exact}} = \begin{cases} \Theta(2^n), & \text{if } \mathrm{D}_{\mathrm{exact}} = \Theta(\frac{2^n}{n}) \\ \Theta(2^n), & \text{if } \mathrm{D}_{\mathrm{exact}} = \Theta(n) \end{cases}$$

Interestingly, Sun et al. [17] also proved a depth lower bound of $\Omega(n)$ for circuits that use arbitrary single-qubit and two-qubit gates, regardless of the number of ancillary qubits from a graph theory perspective. In Sec. 3, we show that our SP+CSP state preparation protocol can also achieve $\Theta(n)$ depth and $\Theta(2^n)$ spacetime allocation. This provides an alternative construction to that of Ref. [18], which is also optimal.

2.4.2 $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set

When compiled into gates from $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ previous methods have achieved [17, 19] the following depth, where a is the number of ancillae available:

$$\mathrm{D}_{\mathrm{approx}} = \begin{cases} O(\frac{2^n \log(2^n/\epsilon)}{n}), & \text{if } a = 0 \\ O(n \log(n) \log(2^n/\epsilon)), & \text{if } a = \Theta(\frac{2^n}{n \log(n)}) \\ O(n \log(n/\epsilon)), & \text{if } a = \Omega(2^n) \end{cases}$$

We can also compute the spacetime allocation cost by multiplying the circuit depth by the number of qubits involved:

$$\mathrm{SA}_{\mathrm{approx}} = \begin{cases} O(2^n \log(2^n/\epsilon)), & \text{if } a = 0 \\ O(2^n \log(2^n/\epsilon)), & \text{if } a = \Theta(\frac{2^n}{n \log(n)}) \\ O(n2^n \log(n/\epsilon)), & \text{if } a = \Omega(2^n) \end{cases}$$

We also note that the method proposed by Yuan et al. [18] will have at least $\Omega(n \log(n/\epsilon))$ depth and $\Omega(2^n \log(n/\epsilon))$ spacetime allocation. This was not stated explicitly in the paper but can be deduced from the fact that single-qubit rotations appear in $O(n)$ different layers and each requires depth at least $\Omega(\log(n/\epsilon))$ in the discrete gate set.

In addition, Clader et al. [20] gave a method with depth $O(\log(2^n/\epsilon))$ and spacetime allocation $O(2^n \log(2^n/\epsilon))$, but under the assumption of unit time FANOUT-CNOT. Decomposing FANOUT-CNOT with 2^n targets into two-qubit CNOT gates incurs a multiplicative overhead of $O(n)$ in the depth of the protocol, which also induces a similar overhead to the spacetime allocation (see Table 1).

In this work, we will show that our method achieves the depth scaling of Ref. [20] in the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set (i.e. not requiring FANOUT-CNOT) while achieving superior spacetime allocation of $O(2^n \log(n/\epsilon))$.

3 SP+CSP State Preparation

In this section, we will walk through the details of our SP+CSP method that achieves $\Theta(N)$ spacetime

allocation while keeping the $\Theta(\log(N))$ depth using the {U(2), CNOT} gate set.

Recall that we wish to create the n-qubit state $|\psi\rangle$ in Eq. (1), which has known coefficients $x_i/\|\mathbf{x}\|$ in the computational basis. We propose to use the following method that first uses a *state preparation* step (**SP**) on a subset of the data qubits, followed by a *controlled state preparation* step (**CSP**). The rationale for doing this rather than a direct SP is that the SP+CSP protocol can harness both the advantages of SP and CSP steps while avoiding their disadvantages if set up correctly. We explain the details of the complexity advantages and disadvantages later in this section at Sec. 3.3.

The general idea of the SP+CSP protocol is to partition the $N = 2^n$ basis states into $M = 2^m$ non-overlapping sets of size $\frac{N}{M}$. Computational basis states for which the first m bits agree (when written in binary) are placed into the same set. Denote the M sets by J_i with $i = 0, \ldots, M-1$. For each i, we let $y_i = \sqrt{\sum_{j\in J_i} |x_j|^2}$.

We first prepare the state of m qubits ($m < n$):

$$U_{\mathrm{SP}} |0^m\rangle = |\phi\rangle = \frac{1}{\|\mathbf{y}\|} \sum_{i=0}^{2^m-1} y_i |i\rangle \tag{5}$$

The total state is now $|\phi\rangle \otimes |0^{n-m}\rangle$.

Next, we perform the controlled state preparation operation U_{CSP}, defined by

$$U_{\mathrm{CSP}}(|i\rangle \otimes |0^{n-m}\rangle) = y_i^{-1} \sum_{j\in J_i} x_j |j\rangle \tag{6}$$

for a particular $|i\rangle$ state being controlled on, so that

$$U_{\mathrm{CSP}}(|\phi\rangle \otimes |0^{n-m}\rangle) = |\psi\rangle = \frac{1}{\|\mathbf{x}\|} \sum_{j=0}^{2^n-1} x_j |j\rangle \tag{7}$$

The circuit diagram is shown in Fig. 6. Implementation details of U_{SP} and U_{CSP} are shown in Fig. 7 and Fig. 8.

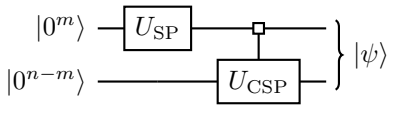

Figure 6: Circuit for the SP+CSP protocol that implements state preparation of an arbitrary n-qubit state $|\psi\rangle$ by first creating an m-qubit state $|\phi\rangle$ and then performing controlled state preparation into the final $n-m$ qubits. The square control ◘ indicates a different operation is performed on the final $n-m$ qubits for each setting of the first m qubits.

We will now describe how to perform the U_{SP} and U_{CSP} circuits.

3.1 SP Circuit Structure

To perform U_{SP}, we give a method that has depth $O(m)$ and uses $O(2^m)$ ancilla qubits. The spacetime allocation is thus at most $O(m2^m)$. If we require a discrete gate set and have approximation error ϵ, this method achieves $O(m+\log(1/\epsilon))$ depth and spacetime allocation $O(m2^m + 2^m \log(m/\epsilon))$.

The idea behind the SP protocol follows previous literature and begins by defining $M-1$ angles that can each be efficiently computed classically from the list of amplitudes $\{y_i\}_{i=0}^{M-1}$. For convenience and consistency with previous literature [8], we use a 2-index angle definition to define the rotation angles:

$$\theta_{s,p} = 2\cos^{-1}\left(\frac{\sqrt{\sum_{l=0}^{2^{m-s-1}-1} |y_{p\cdot 2^{m-s}+l}|^2}}{\sqrt{\sum_{l=0}^{2^{m-s}-1} |y_{p\cdot 2^{m-s}+l}|^2}}\right), \tag{8}$$

where $s = 0, \ldots, m-1$, and $p = 0, \ldots, 2^s - 1$, for a total of $1 + 2 + 4 + \ldots + 2^{m-1} = M - 1$ angles. Naively, computing each angle can require querying as many as $O(2^m)$ entries of the vector $\mathbf{y}$, resulting in $O(2^{2m})$ total classical runtime. However, we can reuse some of the computations by storing the quantities $S_{s,p} = \sum_{l=0}^{2^{m-s}} |y_{p\cdot 2^{m-s}+l}|^2$ in a binary-tree data structure with m levels [3]. The tree has 2^{m-1} leaves; we store the quantities $S_{m-1,p}$ in the leaves for $p = 0, \ldots, 2^{m-1}-1$. The tree is then constructed recursively by the rule that a parent stores the sum of its children. Since $S_{s,p} = S_{s+1,2p} + S_{s+1,2p+1}$, we can verify that the value of the p^{th} node at level s will store the value $S_{s,p}$. The angle $\theta_{s,p}$ can then be determined only from the values stored at this node along with its two children. The total work to construct the entire tree is $O(2^m)$, and the tree has the added benefit that if an individual entry of $\mathbf{y}$ is modified or if the entries arrive in an online fashion, updating the tree data structure can be classically done in time $O(m)$ by following a single path from a leaf back to the root. Additionally, although we do not utilize this property, in the case that $\mathbf{y}$ is sparse and has only d nonzero entries, the tree structure will also be sparse and have at most dm nonzero entries.

This 2-index angle definition will also give us more convenience when labeling the ancilla registers holding the computed angles shown in Eq. (9) and the corresponding data qubits. In particular, s corresponds to the index of the ancilla register A_s (referenced in, e.g., Algorithm 5) as well as the index of the data qubit being processed. Meanwhile, p corresponds to the index of the qubit within the particular ancilla register A_s.

Araujo et al. [24] proposed a low-depth strategy for implementing U_{SP} using $O(M)$ ancilla qubits. The general idea is to "pre-rotate" many angles by prepar-

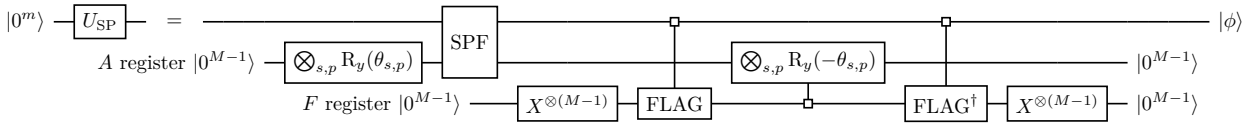

Figure 7: Circuit that implements U_{SP}, which prepares an arbitrary m-qubit state from the state $|0^m\rangle$ with the assistance of $O(2^m)$ ancillae that begin and end in $|0\rangle$. We let $M = 2^m$ and clarify that the controlled rotation gate denotes $M-1$ controlled rotations occurring in parallel, by different angles $\theta_{s,p}$ with $s = 0, \ldots, m-1$ and $p = 0, \ldots, 2^s - 1$. Note that the implementation of SPF and FLAG, given in the appendix, involves $O(M)$ additional unshown ancilla qubits.

ing the states

$$|\theta_{s,p}\rangle = \mathrm{R}_y(\theta_{s,p})\,|0\rangle = \cos\left(\frac{\theta_{s,p}}{2}\right)|0\rangle + \sin\left(\frac{\theta_{s,p}}{2}\right)|1\rangle \tag{9}$$

in parallel for each $s = 0, \ldots, m-1$, $p = 0, \ldots, 2^s - 1$, and then efficiently inject a subset of these states into the data qubits. Which states are injected is controlled by the data qubits themselves, leaving the data qubits entangled with a large garbage register. Clader et al. [20] showed how to uncompute the garbage using a flag mechanism. It was shown that the overall T depth of the injection steps (e.g., SPF and FLAG) was $O(m)$, but the Clifford depth was not studied, and a naive calculation would suggest the Clifford depth is at least $\Omega(m^2)$ when we insist on using only two-qubit Clifford gates, due to the existence of the FANOUT-CNOT gate. In this work, we give an adapted version of Ref. [20] by utilizing a bit-copying mechanism to guarantee that the Clifford depth is also $O(m)$.

The high-level circuit that accomplishes the U_{SP} routine is shown in Fig. 7. The circuit describes the gate sequence that prepares an arbitrary m-qubit state $|\phi\rangle$ with the assistance of $2M - 2 = 2^{m+1} - 2$ ancilla qubits that begin and end[2] in $|0\rangle$. Note that implementations of the SPF circuit and FLAG circuit, discussed in App. A, require an additional $O(M)$ ancillae not shown in the figure to implement the bit copying mechanism mentioned above.

We now follow Ref. [20] and define the action of circuits SPF and FLAG that appear in Fig. 7. First, define the product states

$$\begin{aligned} |\Theta_s\rangle &= \bigotimes_{p=0}^{2^s-1} |\theta_{s,p}\rangle \qquad \text{for } s = 0, 1, \ldots, m-1 \\ |\Theta\rangle &= \bigotimes_{s=0}^{m-1} |\Theta_s\rangle\ . \end{aligned} \tag{10}$$

Next, for each $j = 0, \ldots, M-1$, $s = 0, \ldots, m-1$, $p = 0, \ldots, 2^s - 1$, define

$$f_{(s,p)|j} = \begin{cases} 1 & \text{if } p = j \bmod 2^s \\ 0 & \text{otherwise} \end{cases}\ . \tag{11}$$

[2]Note that the ancillas in the A register of Fig. 7 are only guaranteed to end in $|0\rangle$ when the input to U_{SP} is $|0^m\rangle$. On other input states, this register may end up entangled with the data qubits. See App. B for a discussion on why this feature does not ruin the optimality of our protocol in applications where the input is not always $|0^m\rangle$.

Thus, if we fix a value of j, there are m pairs (s,p) for which $f_{(s,p)|j} = 1$: in fact, for each value of s, there is exactly one $p \in [0, 2^s - 1]$ for which $f_{(s,p)|j} = 1$. The values of (s,p) for which $f_{(s,p)|j} = 1$ correspond to the angle states $|\theta_{s,p}\rangle$ that are injected into the data by the SPF circuit for the data qubit setting $|j\rangle$.

The U_{SP} procedure begins by preparing $|\Theta\rangle$ into an ancilla "angle" register (register A in Fig. 7) of size $M-1$ using $M-1$ parallel R_y gates. The SPF circuit is then applied, which injects some of the angle states into the data and produces the action

$$\mathrm{SPF}\left(|0^m\rangle\,|\Theta\rangle\right) = \frac{1}{\|\mathbf{y}\|}\sum_{j=0}^{M-1} y_j\,|j\rangle\,|g_j\rangle\ . \tag{12}$$

where

$$|g_j\rangle = \bigotimes_{s=0}^{m-1}\bigotimes_{p=0}^{2^s-1} |\theta_{s,p}(1 - f_{(s,p)|j})\rangle\ . \tag{13}$$

Here, the notation $|\theta(1-f)\rangle$ is used to denote that the state is $|\theta\rangle$ when $f = 0$ and $|0\rangle$ when $f = 1$. In other words, the registers holding $|\theta_{s,p}\rangle$ for which $f_{(s,p)|j} = 1$ are replaced with $|0\rangle$. We see that the output of SPF has the correct amplitudes as the target state $|\phi\rangle$, but leaves the data entangled with an $(M-1)$-qubit garbage register.

The FLAG operation computes the bit $f_{(s,p)|j}$ into a fresh ancilla "flag" register[3] (register F in Fig. 7) for each pair (s,p), i.e.,

$$\mathrm{FLAG}(|j\rangle\,|1^{M-1}\rangle) = |j\rangle \bigotimes_{s=0}^{m-1}\bigotimes_{p=0}^{2^s-1} |1 - f_{(s,p)|j}\rangle \tag{14}$$

while leaving the garbage states on the angle qubits untouched. Thus, by applying a controlled-$\mathrm{R}_y(-\theta_{s,p})$ gate controlled by the flag bit $(1 - f_{(s,p)|j})$ onto the angle qubit in the state $|\theta_{s,p}(1 - f_{(s,p)|j})\rangle$ after the FLAG, we reset all of the angle qubits back to $|0\rangle$. This is done in parallel for each pair (s,p). Finally, the adjoint of the FLAG operation (FLAG†) can be applied to uncompute the flag bits and bring the flag register back to $|0^{M-1}\rangle$. The output of this process is the state $|\phi\rangle$ without any garbage.

[3]The need for a flag mechanism following the SPF circuit is the origin of the letter F in the name of the SPF circuit (SP stand for "state preparation") and also in the name of the LOADF circuit (presented in the next section).

The construction for SPF and FLAG described in Ref. [20] was only optimized for the T-depth and T-count. The Clifford count and depth were not explicitly studied. Moreover, it was assumed that a FANOUT-CNOT gate with an arbitrary number of targets could be performed in a single time step. In the appendix, we detail our constructions for the SPF (Sec. A.4) and FLAG (Sec. A.5) subroutines. There, we illustrate how both subroutines can be performed in $O(m)$ depth and $O(m2^m)$ spacetime allocation using $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set and near-optimal costs for $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set.

We document the pseudocode of the SP circuit implementation below in Algorithm 1.

Algorithm 1 SP Circuit (see Fig. 7)

procedure SP(m, D, A, F)
▷ D: data register of size m
▷ A: angle register of size $M-1$
▷ F: flag register of size $M-1$
  for s **in** range(m) **and** p **in** range(2^s) **do**
    Classically compute $\theta_{s,p}$ from $\mathbf{y}$ ▷ Eq. (8)
  end for
  for s **in** range(m) **and** p **in** range(2^s) **do**
    $\mathrm{R}_y(\theta_{s,p}, A_{s,p})$ onto fresh "angle" ancilla
  end for
  SPF(D, A, m) subroutine ▷ $O(m)$
  for s **in** range(m) **and** p **in** range(2^s) **do**
    $\mathrm{X}(F_{s,p})$ onto fresh "flag" ancilla
  end for
  FLAG(D, F, m) subroutine ▷ $O(m)$
  for s **in** range(m) **and** p **in** range(2^s) **do**
    $\mathrm{CR}_y(-\theta_{s,p}, F_{s,p}, A_{s,p})$
  end for
  FLAG$^\dagger(D, F, m)$ subroutine ▷ $O(m)$
  for s **in** range(m) **and** p **in** range(2^s) **do**
    $\mathrm{X}(F_{s,p})$ to reset "flag" ancilla to $|0\rangle$
  end for
end procedure
▷ Total $\mathrm{D}_{\mathrm{exact}}$: $O(m)$
▷ Total $\mathrm{D}_{\mathrm{approx}}$: $O(m + \log(m/\epsilon))$
▷ Total $\mathrm{SA}_{\mathrm{exact}}$: $O(m2^m)$
▷ Total $\mathrm{SA}_{\mathrm{approx}}$: $O(m2^m + 2^m \log(m/\epsilon))$

3.2 CSP Circuit Structure

The controlled state preparation circuit prepares a different $n-m$ qubit state for each of M possible settings $|k\rangle \in \{|0\rangle, \ldots, |M-1\rangle\}$ of an m-qubit control register. Thus, for each k, there are $2^{n-m}-1 = \frac{N}{M}-1$ angles, which we denote by $\theta^{(k)}_{s,p}$, for $s = 0, \ldots, n-m-1$ and $p = 0, \ldots, 2^s-1$. These angles can be computed from the amplitudes of $\mathbf{x}$ by the equation (c.f. Eq. (8))

$$\theta^{(k)}_{s,p} = 2\cos^{-1}\left(\frac{\sqrt{\sum_{l=0}^{2^{n-m-s-1}-1} |x_{k2^{n-m}+p\cdot 2^{n-m-s}+l}|^2}}{\sqrt{\sum_{l=0}^{2^{n-m-s}-1} |x_{k2^{n-m}+p\cdot 2^{n-m-s}+l}|^2}}\right). \tag{15}$$

The total number of angles is then $N-M$. The states $|\theta^{(k)}_{s,p}\rangle$ are then defined as in Eq. (9), and the product states $|\Theta^{(k)}_s\rangle$ and $|\Theta^{(k)}\rangle$ as in Eq. (10). We document the pseudocode of the CSP circuit implementation in Algorithm 2.

Algorithm 2 CSP Circuit (see Fig. 8)

procedure CSP(m, n, D, B, F)
▷ D: data register of size $m-n$
▷ B: buffer register of size $\frac{N}{M}-1$
▷ F: flag register of size $\frac{N}{M}-1$
  for k **in** range(2^m) **do**
    for s **in** range($n-m$) **and** p **in** range(2^s) **do**
      Classically compute $\theta^{(k)}_{s,p}$ from $\mathbf{x}$ ▷ Use Eq. (15)
    end for
  end for
  for s **in** range($n-m$) **and** p **in** range(2^s) **do**
    $\mathrm{X}(F_{s,p})$ onto fresh "flag" ancilla
  end for
  LOADF$(D, B, F)[\theta^{(k)}_{s,p}]$ load angles into buffer ancillae ▷ $O(n)$
  SPF$(D, B, n-m)$ to prepare $n-m$ qubit state with garbage ▷ $O(n-m)$
  FLAG$(D, F, n-m)$ ▷ $O(n-m)$
  LOADF$^\dagger(D, B, F)[\theta^{(k)}_{s,p}]$ to unload angles in buffer ancillae ▷ $O(n)$
  FLAG$^\dagger(D, F, n-m)$ ▷ $O(n-m)$
  for s **in** range($n-m$) **and** p **in** range(2^s) **do**
    $\mathrm{X}(F_{s,p})$ to reset "flag" ancilla
  end for
end procedure
▷ Total $\mathrm{D}_{\mathrm{exact}}$: $O(n)$
▷ Total $\mathrm{D}_{\mathrm{approx}}$: $O(n + \log(n/\epsilon))$
▷ Total $\mathrm{SA}_{\mathrm{exact}}$: $O(2^n)$
▷ Total $\mathrm{SA}_{\mathrm{approx}}$: $O((n-m)2^{n-m} + 2^n \log(n/\epsilon))$

The controlled state preparation circuit, shown in Fig. 8, is a generalized version of the controlled state preparation circuit proposed in Clader et al. [20]. The general idea of the CSP circuit is to first, controlled on the control register being $|k\rangle$, load in the correct state $|\Theta^{(k)}\rangle$. Once this has been done, the same SPF circuit as was used for the U_{SP} protocol is applied to inject the correct angles into the $n-m$ data qubits. A FLAG mechanism similar to that of the U_{SP} protocol is then employed to disentangle the data from the angle register and reset[4] the angle register to $|0^{M-1}\rangle$.

The loading and unloading is accomplished with a circuit we call LOADF, similar to (but not the same as) the LOADF in Ref. [20], which is defined by the

[4] Note that the angle register (B register from Fig. 8) is only guaranteed to end in $|0^{M-1}\rangle$ when the input to U_{CSP} is $|0^{n-m}\rangle$. See Footnote 2 and App. B.

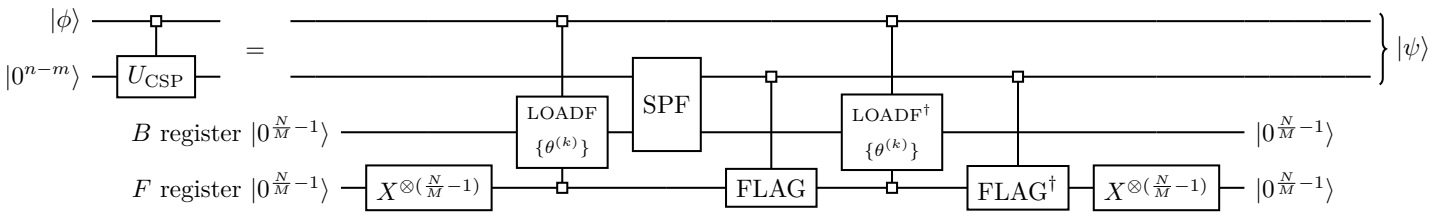

Figure 8: Circuit that implements the U_{CSP} operation, which prepares an arbitrary $n-m$ qubit state for each setting of an m-qubit control register, with the assistance of $O(2^{n-m})$ ancillas that begin and end in $|0\rangle$. We let $M = 2^m$ and $N = 2^n$. Note that implementations of LOADF, SPF, and FLAG, given in the appendix, involve $O(N)$ additional unshown ancillae. We also note that in the actual implementation, one can reduce the first LOADF operation so that it is controlled only by the B register (not on the F register, which is guaranteed to be in the $|1^{N/M-1}\rangle$ state at this stage in the circuit) in order to save a constant depth. We choose to define the LOADF operator in this way only to avoid another definition of the later LOADF†.

action

$$\begin{aligned}&\mathrm{LOADF}\left(|k\rangle\,|0^{N/M-1}\rangle\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\rangle\right]\right)\\ &=|k\rangle\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\theta_{s,p}^{(k)}\rangle\right]\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\rangle\right]\end{aligned} \tag{16}$$

Note that if all the flag bits $f_{s,p}$ are set to 1, the state $|\Theta^{(k)}\rangle$ is prepared into the second register. Our implementation of LOADF is shown in Fig. 19 in the appendix, and involves an additional $O(2^n)$ ancilla qubits.

3.3 Overall depth and spacetime allocation

In this subsection, we compute the overall depth and ancilla allocation of our protocol. To verify the stated complexities of subroutines, we refer the reader to the sections where those subroutines are implemented. We state complexities in the $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set, and then quote the associated complexity in the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set in *(parentheses)* when it is different. We summarize all the individual complexity contributions and the resulted total complexities in Table 3.

Note that to achieve overall error ϵ in state preparation or controlled state preparation, it suffices to prepare each angle state $|\theta_{s,p}\rangle$ to precision ϵ/n. To see this, note that the protocol is equivalent to a sequence of n multi-controlled rotations on the data qubits. If every angle is accurate up to error ϵ/n, each of these n unitaries is enacted up to spectral norm error ϵ/n, yielding total error at most ϵ. More thorough justifications of this appear in [20, Section V B] and [19, Section VIIIA of Supplementary Material].

We begin with U_{SP}, implemented as in Fig. 7. The parallel rotation gates and parallel controlled rotation gates are accomplished in depth $O(1)$ ($\mathrm{D}_{\mathrm{approx}} = O(\log(m/\epsilon))$), while the SPF and FLAG circuits have depth $O(m)$, yielding a total depth $O(m)$ (total[5] $\mathrm{D}_{\mathrm{approx}} = O(m + \log(1/\epsilon)) = O(\log(2^m/\epsilon))$). The required total qubits are the $m + 2M - 2$ that appear in Fig. 7, plus an additional $O(M)$ needed to perform the SPF and FLAG operations. The product of space and depth gives an immediate upper bound of spacetime allocation $O(m2^m)$ ($\mathrm{SA}_{\mathrm{approx}} = O(2^m \log(2^m/\epsilon))$, and indeed for U_{SP} there is a matching lower bound (up to constant factor).

We now consider U_{CSP}. Here, we again note that to perform the operation to ϵ precision, it suffices to perform each single-qubit rotation gate to error ϵ/n. The LOADF operation has depth $O(n)$ ($\mathrm{D}_{\mathrm{approx}} = O(n + \log(1/\epsilon))$. Meanwhile, in this context, the SPF and FLAG operations each have depth $O(n-m)$, yielding total depth $O(n)$ (total depth $O(n+\log(1/\epsilon))$. The required total qubits are the $n + 2N/M$ that appear in Fig. 8, plus the additional $O(N)$ needed to perform the LOADF, SPF, and FLAG circuits. The product of space and depth gives an immediate upper bound of spacetime allocation $O(n2^n)$ ($\mathrm{SA}_{\mathrm{approx}} = O(2^n \log(2^n/\epsilon))$). However, in this case, the actual spacetime allocation is better than this upper bound. The LOADF subroutine achieves spacetime allocation $O(2^n)$ ($\mathrm{SA}_{\mathrm{approx}} = O(2^n \log(n/\epsilon))$) even though it acts on $O(2^n)$ qubits depth $O(n)$ ($\mathrm{D}_{\mathrm{approx}} = O(n + \log(1/\epsilon))$). Meanwhile, FLAG and SPF have spacetime allocation $O((n-m)2^{n-m})$. In total, the procedure has spacetime allocation $O(2^n + (n-m)2^{n-m})$ ($\mathrm{SA}_{\mathrm{approx}} = O(2^n \log(n/\epsilon) + (n-m)2^{n-m})$).

Adding both parts together, the SP+CSP protocol achieves depth $O(n)$ ($\mathrm{D}_{\mathrm{approx}} = O(n + \log(1/\epsilon))$ and has spacetime allocation $O(m2^m + 2^n + (n-m)2^{n-m})$ ($\mathrm{SA}_{\mathrm{approx}} = O(m2^m + 2^n \log(n/\epsilon) + (n-m)2^{n-m})$).

If m is chosen such that

$$\lceil \log_2(n) \rceil \leq m \leq \lfloor n - \log_2(n) \rfloor, \tag{17}$$

we see that the stated result of spacetime allocation $O(2^n)$ ($\mathrm{SA}_{\mathrm{approx}} = O(2^n \log(n/\epsilon))$) is true.

[5] Recall that $O(m + \log(m/\epsilon)) = O(m + \log(1/\epsilon))$, since $\log(m/\epsilon) = \log(m) + \log(1/\epsilon)$, and $O(m + \log(m)) = O(m)$.

	D_{exact}	D_{approx}	# Qubits	SA_{exact}	SA_{approx}
Paralleled Ry	$O(1)$	$O(\log(m/\epsilon))$	$O(2^m)$	$O(2^m)$	$O(2^m \log(m/\epsilon))$
SPF (U_{SP})	$O(m)$	$O(m)$	$O(2^m)$	$O(m2^m)$	$O(m2^m)$
Paralleled X (U_{SP})	$O(1)$	$O(1)$	$O(2^m)$	$O(2^m)$	$O(2^m)$
FLAG (U_{SP})	$O(m)$	$O(m)$	$O(2^m)$	$O(m2^m)$	$O(m2^m)$
Paralleled Control-Ry	$O(1)$	$O(\log(m/\epsilon))$	$O(2^m)$	$O(2^m)$	$O(2^m \log(m/\epsilon))$
Total for U_{SP}	$O(m)$	$O(\log(2^m/\epsilon))$	$O(2^m)$	$O(m2^m)$	$O(2^m \log(2^m/\epsilon))$
Paralleled X (U_{CSP})	$O(1)$	$O(1)$	$O(2^{n-m})$	$O(2^{n-m})$	$O(2^{n-m})$
LOADF	$O(n)$	$O(n+\log(1/\epsilon))$	$O(2^n)$	$O(2^n)$	$O(2^n \log(n/\epsilon))$
SPF (U_{CSP})	$O(n-m)$	$O(n-m)$	$O(2^{n-m})$	$O((n-m)2^{n-m})$	$O((n-m)2^{n-m})$
FLAG (U_{CSP})	$O(n-m)$	$O(n-m)$	$O(2^{n-m})$	$O((n-m)2^{n-m})$	$O((n-m)2^{n-m})$
Total for U_{CSP}	$O(n)$	$O(n+\log(1/\epsilon))$	$O(2^n)$	$O(2^n+(n-m)2^{n-m})$	$O(2^n \log(n/\epsilon)+(n-m)2^{n-m})$
Total	$O(n)$	$O(n+\log(n/\epsilon))$	$O(2^n)$	$O(m2^m+2^n+(n-m)2^{n-m})$	$O(m2^m+2^n \log(n/\epsilon)+(n-m)2^{n-m})$
If Eq. (17) Satisfied	$\boldsymbol{O(n)}$	$\boldsymbol{O(n+\log(n/\epsilon))}$	$\boldsymbol{O(2^n)}$	$\boldsymbol{O(2^n)}$	$\boldsymbol{O(2^n \log(n/\epsilon))}$

Table 3: Individual Complexity Summary for Each Step of State Preparation Protocol. ($M = 2^m$, $N = 2^n$)

In retrospect, we can pinpoint the rationale for pursuing the SP+CSP approach to state preparation. The issue with simply performing the U_{SP} protocol with $n = m$ is that it would require $2^n - 1$ ancilla qubits to store the $2^n - 1$ angles, and each of these ancillae must remain allocated for the entire $O(n)$ depth of the circuit. The SP+CSP protocol gets around this by first preparing the m-qubit state $|\phi\rangle$, requiring only $2^m - 1$ angles, and for each of the 2^m basis states, preparing a different $n - m$ qubit state in the remaining qubits. The latter operation requires $2^{n-m} - 1$ angles (stored in the buffer ancilla register B), so we avoid the need for $O(2^n)$ angle states to be allocated for the entire $O(n)$ depth. The trick comes from the ability to load the correct set of $2^{n-m} - 1$ angles among all $2^n - 2^m$ angles with only $O(2^n)$ spacetime allocation, which is accomplished by our LOADF implementation presented in the appendix.

In the appendix, we also comment on parts of the LOADF subroutine where clean ancillae in the $|0\rangle$ state can be replaced with dirty ancillae in an arbitrary state. In total, a constant fraction of the ancillae in our construction can be dirty, but a constant fraction must remain clean. Future work aims to investigate if a fraction of ancillae that asymptotically approaches 1 can be made dirty. If that is the case, it may be meaningful to differentiate between the portion of spacetime allocation that corresponds to clean ancillae and the fraction that corresponds to dirty ancillae.

4 Preparing Many Copies of Independent Quantum States

This section will show how we manage to utilize the novel encoding method we proposed in Sec. 3 to more efficiently prepare many copies of arbitrary quantum states (same or different ones) using the $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set. Specifically, we consider the task of preparing a product state of w separate N-dimensional states as quickly as possible. If we had $O(wN)$ ancilla qubits, we could perform state preparation on each of the w copies in parallel for total depth $O(\log(N))$. However, in applications, N is likely to be large, and we may not have so many ancillae. Suppose we have only $O(N)$ ancilla qubits—enough to prepare one state in $O(\log(N))$ depth, but not w states. Naively, we could prepare the w states in series, incurring depth $O(w \log(N))$. However, since the spacetime allocation required for a single state preparation is $O(N)$, the spacetime allocation for preparing w states is $O(wN)$, suggesting that if the $O(N)$ ancillae are used reused optimally, we might achieve $O(w)$ total depth, rather than $O(w \log(N))$. Indeed, we now illustrate that using our SP+CSP protocol one can accomplish the task in depth $O(w + \log(N))$, which is constant $O(1)$ depth per copy when $w \geq \Omega(\log(N))$.

There are two essential properties in the single-copy arbitrary state preparation method we described in section 3:

1. All ancilla qubits are disentangled from the data qubits and returned to the $|0\rangle$ state. The previously used ancillae are ready to act as either new data qubits or new ancilla qubits.

2. The amount of spacetime occupied by active ancilla qubits is only $O(N)$, even though the depth is $O(\log(N))$ and there are $O(N)$ ancilla qubits.

Thus, once the ancillae are returned to $|0\rangle$, they can begin to assist with the next state preparation, even if the previous state preparation has not been completed. We can concatenate and stack many of the state preparation circuits one after another.

Quantitatively speaking, we can prepare many unentangled copies of the same quantum state

$$|\psi\rangle^{\otimes w} = \left[\prod_{s=0}^{w-1} U_s\right] |0^n\rangle^{\otimes w} = \left(\frac{1}{\|\mathbf{x}\|}\sum_{i=0}^{2^n-1} x_i\,|i\rangle\right)^{\otimes w} \tag{18}$$

Here U_s's are the same quantum state preparation unitary but acting on different $|0^n\rangle$ states. The above equation does not include ancillas that assist the implementation of the U_s unitaries: even though the U_s's act on different $|0^n\rangle$ data registers, they can share many of the same ancilla qubits.

We can also prepare many copies of different unentangled quantum states

$$\bigotimes_{d=0}^{w-1} |\psi_j\rangle = \left[\prod_{d=0}^{w-1} U_d\right] |0^n\rangle^{\otimes w} = \bigotimes_{d=0}^{w-1}\left(\frac{1}{\|\mathbf{x}^d\|}\sum_{i=0}^{2^n-1} x_i^d\,|i\rangle\right) \tag{19}$$

Here U_d's are *different* quantum state preparation unitaries acting on different $|0^n\rangle$ states and preparing different N-dimensional states described by the vector of coefficients $\mathbf{x}^d$. Even though each U_d is mathematically different, the exact quantum gate scheduling is the same. The only difference is the single-qubit rotation angles.

4.1 SP+CSP Multi-Copy

In order to prepare the joint state in Eq. (19), we can now let $U_d = U_{d(\mathrm{CSP})}U_{d(\mathrm{SP})}$. We first execute all the $U_{d(\mathrm{SP})}$ circuits in parallel and let $m = n - \lceil \log_2(n)\rceil$. That is, we first prepare the state

$$\left[\prod_{d=0}^{w-1} U_{d(\mathrm{SP})}\right] |0^n\rangle^{\otimes w} = \bigotimes_{d=0}^{w-1}(|\phi_d\rangle \otimes |0^{n-m}\rangle), \tag{20}$$

where $|\phi_d\rangle$ is defined from $\mathbf{x}^d$ as in Eq. (5). Then, we jointly execute all the $U_{d(\mathrm{CSP})}$ circuits.

$$\prod_{d=0}^{w-1} U_{d(\mathrm{CSP})} \left(\bigotimes_{d=0}^{w-1}(|\phi_d\rangle \otimes |0^{n-m}\rangle)\right) = \bigotimes_{d=0}^{w-1} |\psi_d\rangle \tag{21}$$

We do not have sufficient ancillae to perform these circuits in parallel, so we perform them with some "indentation" k between each other. That is, we begin the $d = 0$ protocol at layer 0, the $d = 1$ protocol at layer k, the $d = 2$ protocol at layer $2k$, etc.

We illustrate the above operation as a quantum circuit in Fig. 9. If we choose $m = n - \lceil \log_2(n)\rceil$, the first operation described in Eq. (20) can be achieved in $O(\log(N))$ depth and $O(N)$ spacetime allocation. Since each of the $U_{d(\mathrm{SP})}$ is using $\frac{N}{\log(N)}$ qubits, up to $O(\log(N))$ joint states can be parallelized in this way without introducing additional ancilla qubits.

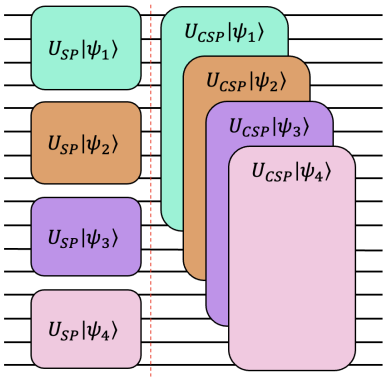

Figure 9: Illustration of stacking different SP+CSP circuit together ($w = 4$)

Now let's walk through the second operation described in Eq. (21). The most ancilla-consuming steps of the CSP circuit are the LOADF operations, and in particular, the layer of doubly-controlled rotation (CCR_y) operations that appear in Fig. 19 of the appendix, which requires $O(N)$ ancillae in the A register. Note that these ancillae can be quickly freed up after the parallelized CCR_y operations by the CopySwap operation under $O(N)$ spacetime allocation, as shown in Fig. 19. Thus we can indent the U_{CSP} part of the later U_d's by some constant number of layers k.

Besides the A register that uses $O(N)$ ancillae, we also have the C, B, and F registers (as shown in Fig. 19) in the U_{CSP} operation. By choosing $m = n - \lceil \log_2(n)\rceil$, we would have negligible ancilla counts for the C register ($m \cdot \frac{N}{M} = m\log(N)$ ancillae), the B register ($M = \frac{N}{\log(N)}$ ancillae), and the F register ($\frac{N}{M} = n$ ancillae). Therefore, they will not contribute much to the overall spacetime allocation cost.

We offer a toy model gate-level illustration in Fig. 10. We can see that the circuit depth will be $2w + \log(N)$ instead of $w\log(N)$. The number of ancillae used is upper bounded by $8N$.

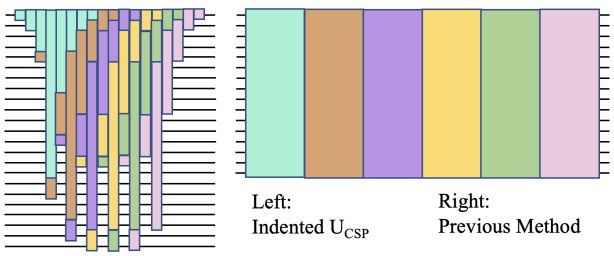

Figure 10: Illustration of stacking *a portion of* the U_{CSP} circuit (LOADF) with indentation $k = 2$.

When the ratio between w and $\log(N)$ grows larger, which is often the case in near-term applications such

as amplitude encoding for quantum machine learning (as illustrated in Sec. 5.1), we would effectively have state preparation circuit depth of $O(w)$, same as the depth using the protocol proposed in [18].

When using the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set, one can develop specific compilation strategies that utilize this early-ancilla-free-up structure based on the particular ϵ values. Asymptotically speaking, since each state preparation occupies spacetime allocation $O(N \log(\log(N)/\epsilon))$, with $O(N)$ ancilla qubits it would be possible to create w states in depth $O(w \log(\log(N)/\epsilon) + \log(N) + \log(1/\epsilon))$.

5 Applications

In general, quantum measurements are probabilistic. Therefore, for many quantum algorithms, we typically need to run many shots with the same quantum circuit setup. That is, we will need to execute the entire circuit (state preparation, algorithm circuit, and measurement) many times to get enough **measurement precision**. Therefore, for many quantum algorithms that require multiple shots/executions and arbitrary quantum state preparation, our novel state preparation method can reduce the average time-per-shot by as much as a factor linear to the number of qubits.

However, there exist more concrete examples where our novel encoding methods can provide advantages beyond the need to improve measurement precision, including but not limited to: quantum machine learning, Hamiltonian simulation, and solving linear systems with algorithms in the style of the Harrow–Hassidim–Lloyd (HHL) algorithm [13].

5.1 Quantum Machine Learning — Batching

One proposed quantum advantage for machine learning tasks is utilizing quantum computers' exponentially large Hilbert space to encode and process data in parallel, providing potential speedup over classical machine learning methods. One of the bottlenecks to realizing such an advantage is that encoding the classical data into the quantum machine in an exponentially compact form (*amplitude encoding* [8]) is costly. In amplitude encoding, we wish to encode the feature vector $\mathbf{x} = [x_0, x_1, ..., x_{N-1}]$ into the amplitude of the quantum state such that

$$|\psi\rangle = \frac{1}{\|\mathbf{x}\|} \sum_{i=0}^{N-1} x_i |i\rangle \tag{22}$$

Notice that $N = 2^n$, where n is the number of required qubits. This indicates we can encode the feature vector in an exponentially compact way.

For instance, if we have a 2×2 image that has a pixel value array of $\mathbf{x} = [232, 31, 62, 137]$ (each pixel has an integer color value ranging from 0 to 255), we only need $\log_2 4 = 2$ qubits to encode the values on the amplitudes:

$$|\psi\rangle = 0.834\,|00\rangle + 0.111\,|01\rangle + 0.223\,|10\rangle + 0.492\,|11\rangle \tag{23}$$

In typical machine learning tasks, a model is trained by feeding it many copies of training data and adjusting the parameters in the model to minimize a loss function evaluated based on that data. One can update the parameters after computing the loss sum of many data points (e.g., batch gradient descent [32]).

In the quantum computing setting, one could execute a QML iteration as follows (also depicted in Fig. 11):

1. encoding many data points into separate quantum states simultaneously;

2. processing all states with, e.g., separate quantum neural network circuits simultaneously; and

3. measuring the output of each circuit simultaneously (disjoint or joint measurements) and computing the total loss.

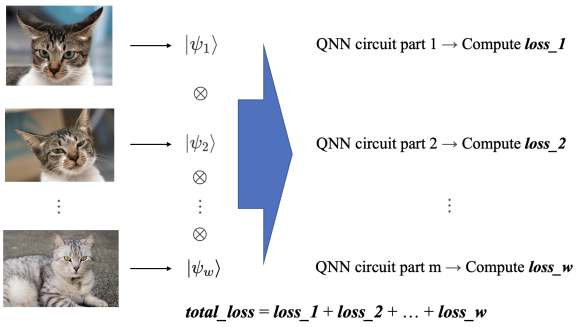

Figure 11: simultaneous execution method example

Since the encoding process can take significant portions of the total required quantum resources, one must consider the best strategy for state preparations.

Using our SP+CSP encoding method, we would be able to achieve $O(w+\log(N))$ circuit depth using only $O(N)$ qubits, using the indentation method described in Sec. 4. In principle, one could encode many data points in effectively *constant* depth, given $O(N)$ ancilla qubits. When running on near-term machines, this effectively matches the best available *theoretical* performance using the protocol proposed by Yuan et al. [18] using the $\{\mathrm{U}(2), \mathrm{CNOT}\}$ gate set. The *actual* best strategy using near-term machines will depend on the machine properties, such as additional available native gates (e.g., CSWAP) and connectivity allowance. For fault-tolerant level quantum machine learning algorithms (e.g., using the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set), our SP+CSP protocol has clear advantages in terms of both depth and spacetime allocation.

5.2 Hamiltonian Simulation — Linear combination of unitaries

Hamiltonian simulation is the task of synthesizing the time evolution unitary $U(t)$ given a length of time t and a description of some Hamiltonian H. For example, Hamiltonian simulation allows one to measure the properties of a time-evolved state

$$|\psi(t)\rangle = U(t)\,|\psi(0)\rangle \tag{24}$$

where $|\psi(0)\rangle$ is the initial state and

$$U(t) = e^{-itH}\,. \tag{25}$$

One method to perform Hamiltonian simulation is to use linear combination of unitaries (LCU) [9, 33]. The idea of this method is to approximate the Hamiltonian evolution operator with a Taylor expansion truncated to order K and then expanded as a linear combination of (unitary) Pauli matrices

$$U(t) \approx \sum_{k=0}^{K} \frac{(-itH)^k}{k!} \tag{26}$$

$$= \sum_{j=0}^{\Gamma-1} \beta_j V_j \tag{27}$$

where V_j are Pauli matrices and the number of Pauli matrices in the linear combination is $\Gamma = \sum_{k=0}^{K} L^k$, where L is the number of Pauli terms in the Hamiltonian H. Let $a = \lceil \log_2(\Gamma) \rceil$.

Implementing the linear combination of unitaries is then done by constructing a SELECT operator and a state-preparation operator B, defined as

$$\text{SELECT} = \sum_{j=0}^{\Gamma-1} |j\rangle\langle j| \otimes V_j \tag{28}$$

$$B\,|0^a\rangle = \frac{\sum_{j=0}^{\Gamma-1} \sqrt{\beta_j}\,|j\rangle}{\sqrt{\sum_{j=0}^{\Gamma-1} \beta_j}}\,. \tag{29}$$

Combining these to form

$$W = (B^\dagger \otimes I)\text{SELECT}(B \otimes I)\,, \tag{30}$$

we see that

$$(\langle 0^a| \otimes I)\,W\,(|0^a\rangle \otimes I) = \frac{\sum_{j=0}^{\Gamma-1} \beta_j V_j}{\sum_{j=0}^{\Gamma-1} \beta_j} \approx \frac{U(t)}{\sum_{j=0}^{\Gamma-1} \beta_j} \tag{31}$$

The above equation is equivalent to the statement that W is an approximate "block-encoding" of $U(t)$ [34, 35], with block-encoding factor $\sum_{j=0}^{\Gamma-1} \beta_j$. Thus, $U(t)$ can be applied by application of W and postselection onto the ancilla state $|0^a\rangle$, which occurs with probability $(\sum_{j=0}^{\Gamma-1} \beta_j)^{-2}$. The success probability can be boosted to 1 by dividing the evolution time t into r segments of length t/r for a specific choice of r, and applying oblivious amplitude amplification [9, 10].

In any case, the state-preparation operator B is a key subroutine of the algorithm, which could be implemented in shallow depth with our state-preparation method. In a typical quantum chemistry simulation setting, the second-quantized Hamiltonian considered includes two-electron and sometimes three-electron integrals, which would make Γ scale as $O(n^4)$ [36] and sometimes $O(n^6)$ [37, 38] (where n indicates the number of orbitals). The number of terms in the LCU will be even larger (depending on the truncation parameter); however, our low-depth state preparation method would have depth scaling as $O(\log(n))$.

Recent works [39–41] have shown the advantage of performing **joint measurements** on multiple copies of quantum states (e.g., $|\psi(t)\rangle$ in Eq. (24)) compared to separate measurements of the end states. Doing this would give us a better sample complexity scaling, which, in the end, will result in better total circuit complexity by saving the total number of circuit executions in order to reach certain measurement thresholds.

5.3 Quantum linear system solving — Repeat until Success

The classical linear system problem is defined as follows: given an $N \times N$ invertible matrix A, and a $N \times 1$ vector $\mathbf{b}$, find the solution vector $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$. The *quantum* linear system problem is to perform the following related task: prepare a state $|\mathbf{x}\rangle = \frac{1}{\|\mathbf{x}\|}\sum_i x_i\,|i\rangle$ whose amplitudes are proportional to the solution vector $\mathbf{x}$. While the state $|\mathbf{x}\rangle$ does not give access to all N entries of the vector $\mathbf{x}$, multiple copies of $|\mathbf{x}\rangle$ can be used, for example, to estimate expectation values $\langle\mathbf{x}|\,M\,|\mathbf{x}\rangle$ of observables M. The first quantum algorithm for solving the quantum linear system problem was by Harrow, Hassidim, and Lloyd (HHL) [13]. Their solution begins by preparing the state $|\mathbf{b}\rangle = \|\mathbf{b}\|^{-1}\sum b_i\,|i\rangle$, where b_i are the entries of the vector $\mathbf{b}$. Then, if A is sparse and, assuming coherent query access to the entries of A in polylog(N) time, the state $|\mathbf{x}\rangle$ can be prepared to high precision in time poly$(\kappa, \log(N))$, where κ is the *condition number* of A, that is, the ratio of the largest to smallest singular values. When $\kappa = O(1)$, this is exponentially faster than classical methods that manipulate vectors of length-N, such as Gaussian elimination or conjugate gradient descent [42]). However, reading out useful information from $|\mathbf{x}\rangle$ can often negate this exponential speedup (leaving the possibility of a polynomial speedup) in specific applications, such as solving differential equations [43]. Even in these cases, there is still a possible exponential advantage in space complexity, due to the exponentially compact representation of the data involved as quantum

states. Relatedly, this compactness is leveraged to yield an unconditional exponential quantum advantage for the linear system problem in the communication complexity setting [44].

It is important to note that even if A is a sparse matrix, the vector $\mathbf{b}$ can often be dense. This is the case, for example, when solving differential equations via mapping to linear systems [43], where $\mathbf{b}$ encodes arbitrary boundary conditions of the problem. Thus, preparing the state $|\mathbf{b}\rangle$ as the first step of the quantum algorithm is precisely an application of our state-preparation protocol.

There are a few reasons that our low-depth state preparation protocol is well suited for usage within algorithms for the quantum linear systems problem. When $\mathbf{b}$ is dense, preparing $\mathbf{b}$ will dominate the algorithm's runtime unless a low-depth state-preparation method is used. Furthermore, extracting information from the solution vector $|\mathbf{x}\rangle$ may require repeating the algorithm many times, and thus preparing many copies of $|\mathbf{b}\rangle$; as outlined in Sec. 4, the fact that our method has optimal spacetime allocation allows many copies to be rapidly prepared with efficient ancilla usage. Furthermore, depending on which quantum approach is taken, the algorithm itself may require preparing many copies of $|\mathbf{b}\rangle$ to generate a single copy of $|\mathbf{x}\rangle$. For example, in modern approaches such as [34], one can access the matrix A, via a unitary "block-encoding" U_A. Using signal processing techniques, one can create a block-encoding $U_{A^{-1}}$ of A^{-1}. The unitary $U_{A^{-1}}$ uses $\tilde{O}(\kappa)$ calls to the unitary U_A, and performs the operation (up to small error)

$$U_{A^{-1}} |\mathbf{b}\rangle |0^\ell\rangle \approx c |\mathbf{x}\rangle |0^\ell\rangle + |\perp\rangle \tag{32}$$

where $|\perp\rangle$ is in the null space of the projector $I \otimes |0^\ell\rangle \langle 0^\ell|$, and $|c| = O(1/\kappa)$. A measurement of the ancilla register then produces the desired state $|\mathbf{x}\rangle$ with probability $|c|^2 = O(1/\kappa^2)$. Thus, to produce $|\mathbf{x}\rangle$, it suffices to create $O(\kappa^2)$ copies of $|\mathbf{b}\rangle$ and make $\tilde{O}(\kappa)$ queries to U_A for each copy, for a total of $\tilde{O}(\kappa^3)$ queries. In principle, these copies could be processed in parallel, and, as described above, our method offers a minimal-depth approach for preparing $|\mathbf{b}\rangle^{\otimes(O(\kappa^2))}$ with only $O(N)$ ancillae. Using amplitude amplification, the number of queries to U_A can be reduced to $\tilde{O}(\kappa^2)$, which would involve $O(\kappa)$ *serial* reflections about the state $|\mathbf{b}\rangle$, an operation that can be performed optimally by our protocol (see App. B). Finally, using variable-time amplitude amplification [14, 34] or ideas from adiabatic quantum computing [45, 46], the U_A query complexity can be further reduced to $O(\kappa)$. This process also requires $O(\kappa)$ serial applications of our state-preparation protocol.

5.4 Block-encodings

We have already seen above how block-encodings are useful as a framework for giving a quantum algorithm, such as Hamiltonian simulation or quantum linear systems solvers, access to some underlying data [34, 35, 47]. Quantum state preparation is also a useful ingredient for constructing block-encodings.

The definition of a block-encoding of a matrix A is a unitary U_A where A appears in the upper left block, scaled by some constant α such that $\|A/\alpha\| \leq 1$:

$$U_A = \begin{bmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{bmatrix}. \tag{33}$$

For example, given a Hamiltonian H representing some physical system, one may seek a block-encoding U_H of H, which can then be used to build algorithms that produce the ground state or thermal state of H. It can also be used to perform Hamiltonian simulation via qubitization [12, 35], a technique that can have superior performance to the LCU method discussed previously.

How the block-encoding U_A is constructed depends on the contents of A and the context of the algorithm. Many of the most common methods explicitly involve the need for QSP or controlled QSP (see Sec. 4.2 of [35]).

For example, when A is a Gram matrix—that is, a matrix for which entries $A_{ij} = \langle\psi_i|\phi_j\rangle$ for some sets $\{|\psi_i\rangle\}_i$, $\{|\phi_j\rangle\}_j$—one can provide a block encoding of A as $U_A = U_L^\dagger U_R$, where U_L and U_R are unitaries that prepare the arbitrary states $|\psi_i\rangle$ and $|\phi_j\rangle$ [35, Lemma 47]). A similar approach can be used to construct block-encodings of arbitrary matrices [20, 34].

Both U_L and U_R can be constructed using our controlled state preparation protocol with the optimized depth of $O(\log(N))$ and $O(N)$ ancilla qubits.

6 Code Availability

We present two gate-level implementation examples at https://github.com/guikaiwen/QSP_Paper_Artifact, including:

- a 2 + 2 case ($m = 2$, $n = 4$) case without COPY
- a 1 + 2 case ($m = 1$, $n = 3$) case with COPY

Acknowledgments

We thank Connor Hann, Yunong Shi, Erhan Bas, Pei Zeng, Ming Yuan, Siteng Kang, Kyle Brubaker, Martin Schuetz, Fei Chen, Liang Jiang, Eric Kessler, Senrui Chen, Sisi Zhou, and Gideon Lee for their helpful discussions. This work is funded in part by EPiQC, an NSF Expedition in Computing, under award CCF-1730449. FTC is Chief Scientist for Quantum Software at Infleqtion and an advisor to Quantum Circuits, Inc.

References

[1] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. "Quantum machine learning". Nature **549**, 195–202 (2017).

[2] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". Nature Physics **10**, 631–633 (2014).

[3] Iordanis Kerenidis and Anupam Prakash. "Quantum recommendation systems". In 8th Innovations in Theoretical Computer Science Conference (ITCS 2017). Volume 67 of Leibniz International Proceedings in Informatics (LIPIcs), pages 49:1–49:21. (2017).

[4] Patrick Rebentrost, Adrian Steffens, Iman Marvian, and Seth Lloyd. "Quantum singular-value decomposition of nonsparse low-rank matrices". Physical review A **97**, 012327 (2018).

[5] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. "q-means: A quantum algorithm for unsupervised machine learning". Advances in neural information processing systems (2019).

[6] Iordanis Kerenidis and Jonas Landman. "Quantum spectral clustering". Physical Review A **103**, 042415 (2021).

[7] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification". Physical review letters **113**, 130503 (2014).

[8] Maria Schuld and Francesco Petruccione. "Machine learning with quantum computers". Springer. (2021).

[9] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. "Simulating Hamiltonian dynamics with a truncated Taylor series". Physical review letters **114**, 090502 (2015).

[10] Dominic W Berry, Andrew M Childs, and Robin Kothari. "Hamiltonian simulation with nearly optimal dependence on all parameters". In 2015 IEEE 56th annual symposium on foundations of computer science. Pages 792–809. IEEE (2015).

[11] Guang Hao Low and Isaac L Chuang. "Optimal Hamiltonian simulation by quantum signal processing". Physical review letters **118**, 010501 (2017).

[12] Guang Hao Low and Isaac L Chuang. "Hamiltonian simulation by qubitization". Quantum **3**, 163 (2019).

[13] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". Physical review letters **103**, 150502 (2009).

[14] Andris Ambainis. "Variable time amplitude amplification and quantum algorithms for linear algebra problems". In STACS'12 (29th Symposium on Theoretical Aspects of Computer Science). Volume 14, pages 636–647. LIPIcs (2012).

[15] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. "Quantum linear system algorithm for dense matrices". Physical review letters **120**, 050502 (2018).

[16] Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer. "Trading T-gates for dirty qubits in state preparation and unitary synthesis". arXiv.1812.00954 (2018).

[17] Xiaoming Sun, Guojing Tian, Shuai Yang, Pei Yuan, and Shengyu Zhang. "Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2023).

[18] Pei Yuan and Shengyu Zhang. "Optimal (controlled) quantum state preparation and improved unitary synthesis by quantum circuits with any number of ancillary qubits". Quantum **7**, 956 (2023).

[19] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. "Quantum state preparation with optimal circuit depth: Implementations and applications". Physical Review Letters **129**, 230504 (2022).

[20] B David Clader, Alexander M Dalzell, Nikitas Stamatopoulos, Grant Salton, Mario Berta, and William J Zeng. "Quantum resources required to block-encode a matrix of classical data". IEEE Transactions on Quantum Engineering (2023).

[21] Gregory Rosenthal. "Query and depth upper bounds for quantum unitaries via grover search". arXiv.2111.07992 (2021).

[22] Neil J. Ross and Peter Selinger. "Optimal ancilla-free Clifford+T approximation of z-rotations". Quantum Info. Comput. (2016).

[23] Ryan Babbush, Craig Gidney, Dominic W Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. "Encoding electronic spectra in quantum circuits with linear T complexity". Physical Review X **8**, 041015 (2018).

[24] Israel F Araujo, Daniel K Park, Francesco Petruccione, and Adenilton J da Silva. "A divide-and-conquer algorithm for quantum state preparation". Scientific reports **11**, 1–12 (2021).

[25] Vivek V. Shende and Igor L. Markov. "On the CNOT-cost of TOFFOLI gates". Quantum Info. Comput. (2009).

[26] John A Smolin and David P DiVincenzo. "Five two-bit quantum gates are sufficient to implement the quantum Fredkin gate". Physical Review A **53**, 2855 (1996).

[27] Edward Walker. "The real cost of a CPU hour". Computer **42**, 35–41 (2009).

[28] Yongshan Ding, Xin-Chuan Wu, Adam Holmes, Ash Wiseth, Diana Franklin, Margaret Martonosi, and Frederic T Chong.

"Square: Strategic quantum ancilla reuse for modular quantum programs via cost-effective uncomputation". In 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). Pages 570–583. IEEE (2020).

[29] Martin Plesch and Časlav Brukner. "Quantum-state preparation with universal gate decompositions". Phys. Rev. A **83**, 032302 (2011).

[30] Xiao-Ming Zhang and Xiao Yuan. "On circuit complexity of quantum access models for encoding classical data". arXiv.2311.11365 (2023).

[31] Michael A Nielsen and Isaac Chuang. "Quantum computation and quantum information". American Association of Physics Teachers. (2002).

[32] Sebastian Ruder. "An overview of gradient descent optimization algorithms". arXiv.1609.04747 (2016).

[33] Andrew M Childs, Dmitri Maslov, Yunseong Nam, Neil J Ross, and Yuan Su. "Toward the first quantum simulation with quantum speedup". Proceedings of the National Academy of Sciences **115**, 9456–9461 (2018).

[34] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. "The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation". In Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP). Pages 33:1–33:14. (2019).

[35] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. "Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics". In Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC). Pages 193–204. (2019).

[36] Trygve Helgaker, Poul Jorgensen, and Jeppe Olsen. "Molecular electronic-structure theory". John Wiley & Sons. (2013).

[37] Mario Motta, Tanvi P Gujarati, Julia E Rice, Ashutosh Kumar, Conner Masteran, Joseph A Latone, Eunseok Lee, Edward F Valeev, and Tyler Y Takeshita. "Quantum simulation of electronic structure with a transcorrelated Hamiltonian: improved accuracy with a smaller footprint on the quantum computer". Physical Chemistry Chemical Physics **22**, 24270–24281 (2020).

[38] Sam McArdle and David P Tew. "Improving the accuracy of quantum computational chemistry using the transcorrelated method". arXiv.2006.11181 (2020).

[39] Sebastien Bubeck, Sitan Chen, and Jerry Li. "Entanglement is necessary for optimal quantum property testing". In 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS). Pages 692–703. IEEE (2020).

[40] Sitan Chen, Jordan Cotler, Hsin-Yuan Huang, and Jerry Li. "Exponential separations between learning with and without quantum memory". In 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS). Pages 574–585. IEEE (2022).

[41] Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, et al. "Quantum advantage in learning from experiments". Science **376**, 1182–1186 (2022).

[42] Jonathan Richard Shewchuk et al. "An introduction to the conjugate gradient method without the agonizing pain". 1994 Technical Report (1994).

[43] Ashley Montanaro and Sam Pallister. "Quantum algorithms and the finite element method". Physical Review A **93**, 032324 (2016).

[44] Ashley Montanaro and Changpeng Shao. "Quantum communication complexity of linear regression". ACM Trans. Comput. Theory (2023).

[45] Yiğit Subaşı, Rolando D. Somma, and Davide Orsucci. "Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing". Phys. Rev. Lett. **122**, 060504 (2019).

[46] Pedro CS Costa, Dong An, Yuval R Sanders, Yuan Su, Ryan Babbush, and Dominic W Berry. "Optimal scaling quantum linear-systems solver via discrete adiabatic theorem". PRX Quantum **3**, 040303 (2022).

[47] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. "Grand unification of quantum algorithms". PRX Quantum **2**, 040203 (2021).

[48] Craig Gidney. "Quirk: A drag-and-drop quantum circuit simulator". `https://algassert.com/quirk` (2016).

[49] Alexander M Dalzell, B David Clader, Grant Salton, Mario Berta, Cedric Yen-Yu Lin, David A Bader, Nikitas Stamatopoulos, Martin JA Schuetz, Fernando GSL Brandão, Helmut G Katzgraber, et al. "End-to-end resource analysis for quantum interior-point methods and portfolio optimization". PRX Quantum **4**, 040325 (2023).

A Circuit implementations

A.1 Qubit Data Copying

Our implementation of SPF and FLAG circuits will require additional ancillae that are used as part of coherent data-copying subroutines. That is, we perform the isometry, which was also utilized, for example, in Refs. [17, 18]

$$\alpha\,|0\rangle + \beta\,|1\rangle \mapsto \alpha\,|0^c\rangle + \beta\,|1^c\rangle \tag{34}$$

using only CNOT gates and in depth $\lceil \log_2(c) \rceil$. We will need this subroutine because we often want to perform many CSWAP gates with disjoint targets but controlled by the same register. We can accomplish this in shallower depth by copying the control register and applying the CSWAPs in parallel. Clader et al. [20] avoided this issue by assuming the ability to do FANOUT-CNOT with an arbitrary number of targets, which is a Clifford gate, in a single time step.

The protocol for copying consists of $\lceil \log_2(c) \rceil$ sequential depth-1 operations, labeled by $\bigoplus_0, \ldots, \bigoplus_{\lceil \log_2(c) \rceil - 1}$. We refer to these as *copying layers*. In particular, $\bigoplus_j$ consists of a single layer of 2^j parallel CNOTs where the targets of the CNOTs are fresh ancillae. This is illustrated in Fig. 12 and described in Algorithm 3. Notably, while the total number of qubits is c and the depth is $\lceil \log_2(c) \rceil$, most of the ancillae need not be allocated until close to the end of the protocol, and thus the spacetime allocation is $\Theta(c)$, rather than $O(c \log(c))$.

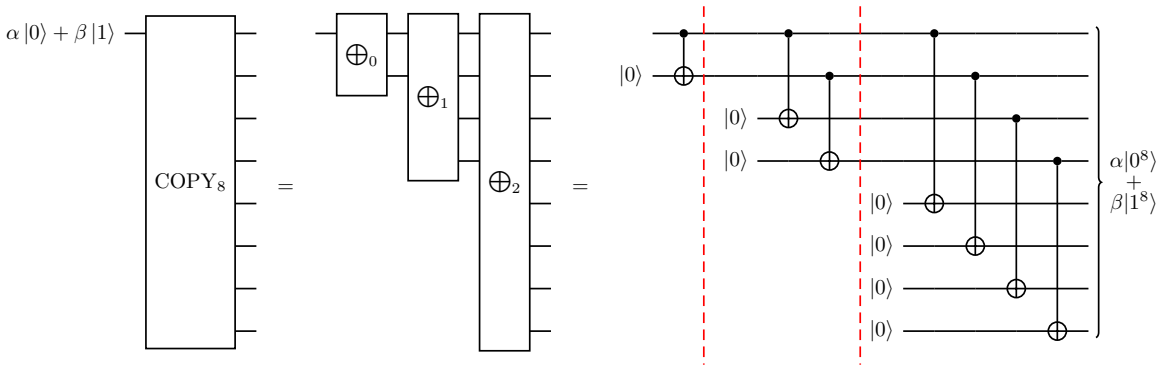

Figure 12: Example circuit implementing state copy subroutine for $c = 8$ qubits. An arbitrary single-qubit state $\alpha\,|0\rangle + \beta\,|1\rangle$ is copied into $c = 2^t$ qubits (in the sense of mapping to $\alpha\,|0^c\rangle + \beta\,|1^c\rangle$) via a series of t layers of CNOTs, denoted $\bigoplus_0, \ldots, \bigoplus_{t-1}$. The $c - 1$ qubits are fresh ancilla qubits that begin in the state $|0\rangle$. The depth is $\log_2(c)$, and the spacetime allocation is only c, as seen in the right diagram, by waiting to allocate ancillae until the final moment that they are needed.

Algorithm 3 State-Copy Subroutine

procedure COPY$_c(R)$
  ▷ R is a register of size c, where c is assumed to be the power of 2 for simplicity
  for i **in** range($\log_2(c)$) **do** ▷ Each value of i occupies $O(1)$ depth
    $\bigoplus_i(R)$
  end for
end procedure
procedure $\bigoplus_t(R)$
  for j in range(2^t) **do** ▷ All values of j performed in parallel
    CNOT($R_{jc2^{-t}}$, $R_{jc2^{-t}+c2^{-t-1}}$)
  end for
end procedure

▷ Total D$_{\text{exact}}$: $\log_2(c)$
▷ Total D$_{\text{approx}}$: $\log_2(c)$
▷ Total SA$_{\text{exact}}$: $2c - 2$
▷ Total SA$_{\text{approx}}$: $2c - 2$

A.2 Parallel CSWAP

With the help of the State-Copy subroutine, we can now effectively perform the parallel CSWAP operations using single-qubit gates and two-qubit CNOT gates, without using the FANOUT-CNOT gate that was required in Ref. [20] to perform the parallel CSWAP when sharing the same control bits.

We denote a layer of 2^t parallel CSWAPs by CS_t, as depicted in Fig. 13 and Algorithm 4.

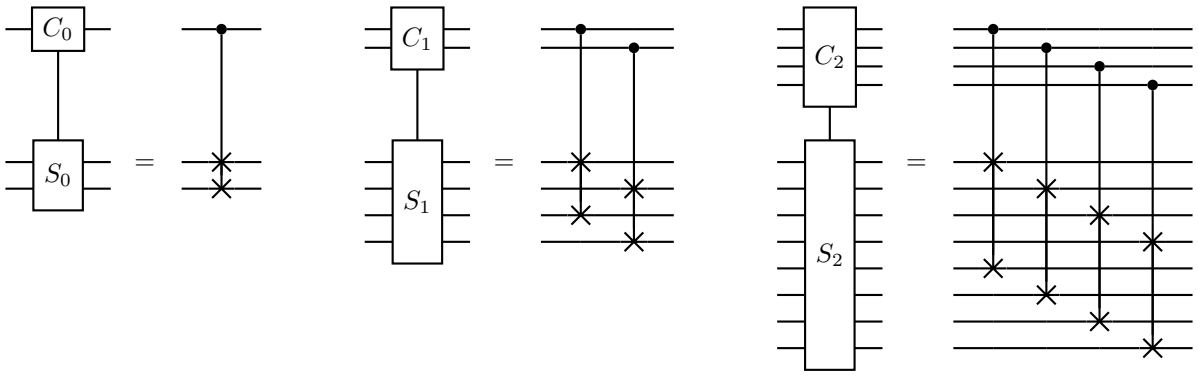

Figure 13: Implementation of CS_t for $t = 0, 1, 2$, which can be accomplished in one layer of parallel CSWAP gates.

Algorithm 4 Parallel CSWAP

1: **procedure** $\mathrm{CS}_t(R, S)$
▷ t: log number of parallel control-swaps, R: control bit data register with at least 2^t qubits, S: target bit angle register with at least 2^{t+1} qubits (note that the subscript here labels the qubit indices of every *single* register)
2: **for** i in range(2^t) **do** ▷ All values of i performed in parallel
3: CSWAP(R_i, S_i, S_{i+2^t})
4: **end for**
5: **end procedure**

A.3 COPY Layer Application Example

In Fig. 14, we illustrate how we can effectively parallelize many CSWAP gates with the same control bits and different target bits with *constant* ancilla and total depth overhead. The same logic can also apply to Toffoli gates.

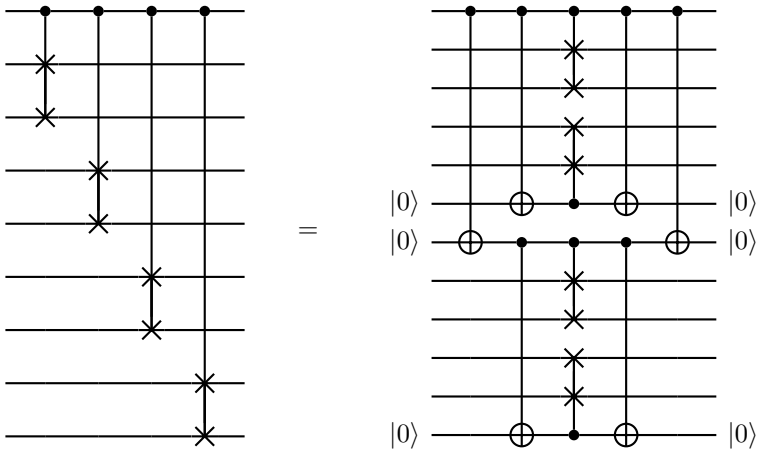

Figure 14: Circuit equivalence with and without COPY layers: note that we are only introducing an additional $O(\log(N))$ CNOT layers and $O(N)$ ancilla qubits in the $|0\rangle$ state, same scale as the total circuit depth and total number of ancilla qubits.

A.4 SPF

Now we describe the implementation of the SPF circuit. The circuit acts on $m + M - 1$ qubits, and our implementation uses additional $M/2 - m$ ancillae that begin in and end in $|0\rangle$. The circuit is similar to that described in Ref. [20], but in our implementation, all gates are two-qubit gates. The idea is to enact $\mathrm{R}_y(\theta_{s,p})$ rotations onto the m data qubit registers in sequence, but where which pair (s, p) is used for the sth rotation depends on the value of the first $s - 1$ registers. We assume that each of the $|\theta_{s,p}\rangle$ states have already been prepared. The rotations are enacted by, for each $s = 0, \ldots, m - 1$, swapping ("injecting") the correct $|\theta_{s,p}\rangle$ state with the sth data qubit. Exactly one $\theta_{s,p}$ will be injected for each s. Since $(0, 0)$ is the only (s, p) with $s = 0$, the first step is to swap $|\theta_{0,0}\rangle$ with the first data qubit. The second step is to, conditioned on the first data register being $|1\rangle$, swap the registers holding $|\theta_{1,0}\rangle$ and $|\theta_{1,1}\rangle$, and then inject the register originally holding $|\theta_{1,0}\rangle$ into the second data register (using a swap gate). In general, the sth rotation is enacted by first swapping the correct $|\theta_{s,p}\rangle$ to the top of the size-2^s register initially holding the states $|\theta_{s,p}\rangle$, and then swapping the top register into the sth data qubit. Naively, this would seem to require at least $O(m^2)$ depth, as there are m rotations, and deciding which qubit to inject for each requires $O(m)$ rounds of controlled-swapping. Ref. [20] observed that this can be reduced to $O(m)$ depth, assuming access to FANOUT-CNOT with $O(M)$ targets that act in unit time. To decompose such a FANOUT-CNOT gate into two-qubit gates, at least $O(m)$ depth would typically be required; thus, the depth of the Clader et al. [20] SPF implementation in the $\{U(2), \mathrm{CNOT}\}$ gate set would be $O(m^2)$, not $O(m)$.

We give a circuit compilation method for reducing the depth back to $O(m)$ using a few extra ancillae. We do so by interspersing the CSWAPs with copying layers. In particular, for each $u = s + 1, s + 2, \ldots, m - 1$, data qubit s is needed to control 2^{u-1-s} CSWAPs on the angle register that starts in state $|\Theta_u\rangle$. Thus, once we have applied $u - 1 - s$ copying layers to it, we are free to perform the CSWAPs in parallel; we denote the operation that performs 2^t CSWAPs in parallel by CS_t, following Fig. 13. While we apply CSWAPs controlled on data qubit s, we are free to copy other data qubits in parallel. Overall, we manage to perform the operation in only $O(m)$ depth.

We describe the SPF circuit in several ways. In Fig. 15, we give a complete example of SPF for $m = 4$, illustrating how ancilla qubits are used to host copies of the data qubits for the purpose of controlling swaps. However, it is hard to fully see the pattern for small values of m. In Fig. 16, we show the first half of the SPF circuit for a larger $m = 7$ example and label the various registers. In this figure, one can easily verify that the depth is $O(m)$. Finally, in Algorithm 5, we give pseudocode for the implementation of SPF, consistent with the labels appearing in Fig. 16. To understand the idea behind the circuits, we now describe several rules that must be obeyed (and the reasoning why):

1. $\bigoplus_j(D_q)$ needs to happen after $\bigoplus_i(D_q)$ if $j < i$, $\forall q$:
    - This is true by construction, shown in Fig. 12.
2. $\bigoplus_j(D_q)$ needs to happen after $\mathrm{SWAP}(D_{q,0}, A_{q,0})$, $\forall j$, $\forall q$:
    - This is because the injection of the angle state into data qubit q must occur before we can begin to copy data qubit q.
3. $\mathrm{CS}_k(D_q, \cdot)$ needs to happen after $\bigoplus_{k-1}(D_q)$, $\forall q$:
    - This makes sure we can have enough qubits for the CSWAP sequence to control on in order to guarantee maximal parallelization.
4. $\mathrm{CS}_j(A_q)$ needs to iterate through all j values in the order of $q - 1$ to 0 before $\mathrm{SWAP}(D_{q,0}, A_{q,0})$, $\forall q$:
    - This makes sure all the values are swapped into the correct place for the current qubit based on the previous qubits' amplitudes before the current qubit is pumped into the data register.

Because of the existence of criteria #2 and #4, we cannot simply execute the STATE-COPY subroutine at one time from each D_q as we do in the FLAG circuit compilation presented in the next subsection (algorithm 6). Doing that would result in $O(m^2)$ depth. Therefore, we have to intersperse the CSWAP sequences with the copying layers.

The total number of copying layers of data qubits j that we need is $m-2-j$ (for $j = 0, \ldots, m-3$). Data qubits $m-2$ and $m-1$ need not be copied. Thus, the total number of ancillae needed is $\sum_{j=0}^{m-3}(2^{m-2-j}-1) = 2^{m-1}-m$.

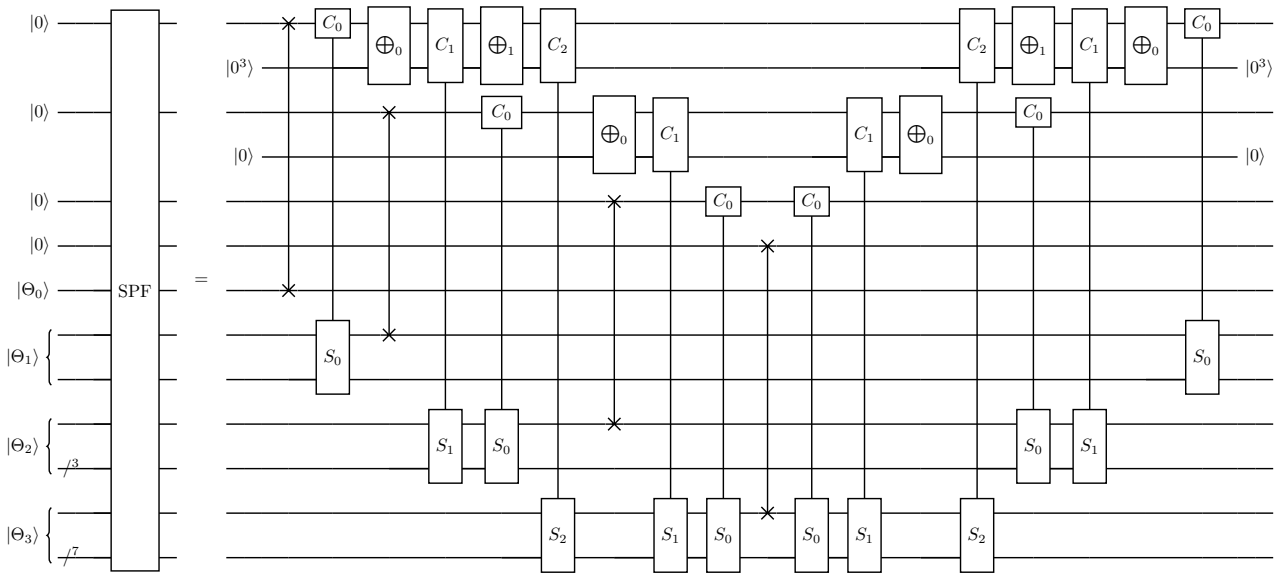

Figure 15: SPF circuit for preparing a state with $m = 4$ qubits, which are left entangled with $2^q - 1$ qubits (garbage). The circuit has depth $O(m)$ and uses an additional $O(2^m)$ ancillae that begin and end in $|0\rangle$. Thus, the total spacetime allocation is upper bounded by $O(m2^m)$. The second half of the SPF circuit is very similar to the reverse of the first half, except that SWAP gates are not present.

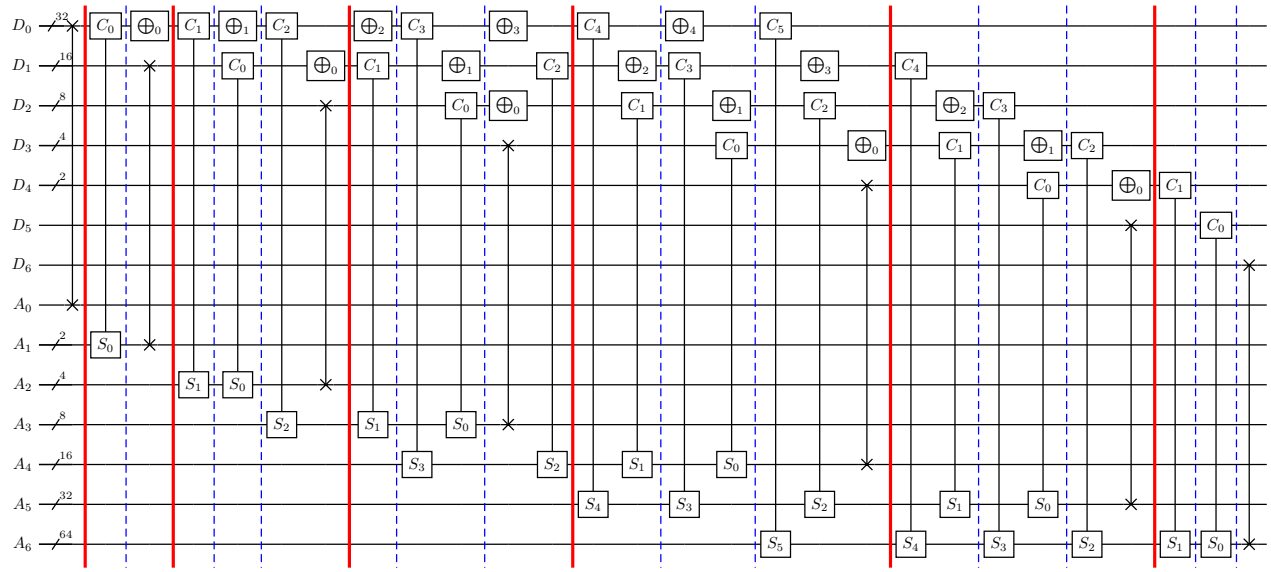

Figure 16: First part of SPF for $m = 7$. The circuit consists of 7 data registers (D_0 - D_6) and 7 ancilla registers (A_0 - A_6). Each space between the red line in the circuit represents 1 iteration of the i values in algorithm 5. There are 3 parallelized gate sequences in each red line space, separated by the blue dashed line. The first parallelized gate sequence corresponds to the first for loop in line 3 - line 9. The second parallelized gate sequence corresponds to the second for loop in line 10 - line 16. The third parallelized gate sequence corresponds to the third for loop in line 17 - line 25. (Notice that we might not have all 3 sequences in all of the boxes based on the condition, for instance, at the beginning of the circuit)

Algorithm 5 SPF Subroutine

procedure SPF(D, A, m)
▷ D is the data register of size m
▷ $A = A_0, \ldots, A_{m-1}$ is the angle register where A_j is size 2^j
  for i **in** range(m) **do** ▷ Each value of i occupies $O(1)$ depth, shown as one box in Fig. 16
    for q **in** range($i-1$) **do** ▷ 1st parallelized sequence in the space between red lines in Fig. 16.
      if $(i-q)$ is odd **and** $\frac{3(i-q-1)}{2} - 1 \leq m-q-3$ **then**
        $\bigoplus_{\frac{3(i-q-1)}{2}-1}(D_q)$
      else if $(i-q)$ is even **and** $\frac{3(i-q)}{2} - 2 \leq m-q-2$ **then**
        $\mathrm{CS}_{\frac{3(i-q)}{2}-2}(D_q, A_{i-1+\frac{i-q}{2}})$
      end if
    end for
    for q **in** range(i) **do** ▷ 2nd parallelized sequence in the space between red lines in Fig. 16.
      if $(i-q)$ is odd **and** $\frac{3(i-q-1)}{2} \leq m-q-2$ **then**
        $\mathrm{CS}_{\frac{3(i-q-1)}{2}}(D_q, A_{i+\frac{i-q-1}{2}})$
      else if $(i-q)$ is even **and** $\frac{3(i-q)}{2} - 2 \leq m-q-3$ **then**
        $\bigoplus(\frac{3(i-q)}{2} - 2, D_q)$
      end if
    end for
    for q **in** range($i+1$) **do** ▷ 3rd parallelized sequence in the space between red lines in Fig. 16.
      if $i == q$ **then**
        SWAP$(D_{q,0}, A_{i,0})$
      else if $(i-q)$ is odd **and** $\frac{3(i-q-1)}{2} \leq m-q-3$ **then**
        $\bigoplus_{\frac{3(i-q-1)}{2}}(D_q)$
      else if $(i-q)$ is even **and** $\frac{3(i-q)}{2} - 1 \leq m-q-2$ **then**
        $\mathrm{CS}_{\frac{3(i-q)}{2}-1}(D_q, A_{i+\frac{i-q}{2}})$
      end if
    end for
  end for
  for i **in** reversed(range($1, m$)) **do** ▷ Each value of i occupies $O(1)$ depth, run in reversed order
    for q **in** range($i+1$) **do** ▷ All values of q performed in parallel
      if $i == q$ **then**
        continue
      else if $(i-q)$ is odd **and** $\frac{3(i-q-1)}{2} \leq m-q-3$ **then**
        $\bigoplus_{\frac{3(i-q-1)}{2}}(D_q)$
      else if $(i-q)$ is even **and** $\frac{3(i-q)}{2} \leq m-q-2$ **then**
        $\mathrm{CS}_{\frac{3(i-q)}{2}-1}(D_q, A_{i+\frac{i-q}{2}})$
      end if
    end for
    for q **in** range(i) **do** ▷ All values of q performed in parallel
      if $(i-q)$ is odd **and** $\frac{3(i-q-1)}{2} \leq m-q-2$ **then**
        $\mathrm{CS}_{\frac{3(i-q-1)}{2}}(D_q, A_{i+\frac{i-q-1}{2}})$
      else if $(i-q)$ is even **and** $\frac{3(i-q)}{2} - 2 \leq m-q-3$ **then**
        $\bigoplus_{\frac{3(i-q)}{2}-2}(D_q)$
      end if
    end for
    for q **in** range($i-1$) **do** ▷ All values of q performed in parallel
      if $(i-q)$ is odd **and** $\frac{3(i-q-1)}{2} - 1 \leq m-q-3$ **then**
        $\bigoplus_{\frac{3(i-q-1)}{2}-1}(D_q)$
      else if $(i-q)$ is even **and** $\frac{3(i-q)}{2} - 2 \leq m-q-2$ **then**
        $\mathrm{CS}_{\frac{3(i-q)}{2}-2}(D_q, A_{i-1+\frac{i-q}{2}})$
      end if ▷ Total $\mathrm{D}_{\mathrm{exact}}$: $O(m)$
    end for ▷ Total $\mathrm{D}_{\mathrm{approx}}$: $O(m)$
  end for ▷ Total $\mathrm{SA}_{\mathrm{exact}}$: $O(m2^m)$
end procedure ▷ Total $\mathrm{SA}_{\mathrm{approx}}$: $O(m2^m)$

A.5 FLAG

The implementation of the FLAG circuit is similar to that of the SPF circuit, with a few simplifications. Here, the data qubits are only acting as controls, and there is no injection of angles into the data qubits. Thus, we do not need to alternate between copying layers and CSWAP layers; we can simply perform all the copying at the beginning, then all the CSWAPs, and then all the uncopying. The conceptual idea behind the FLAG implementation is that a flag is set in the p^{th} flag qubit by first flagging qubit 0 and then swapping qubit 0 into the p^{th} position using a sequence of CSWAP layers with different data qubits acting as the control. We give an example of FLAG for $m = 4$ in Fig. 17 and pseudocode for FLAG in Algorithm 6.

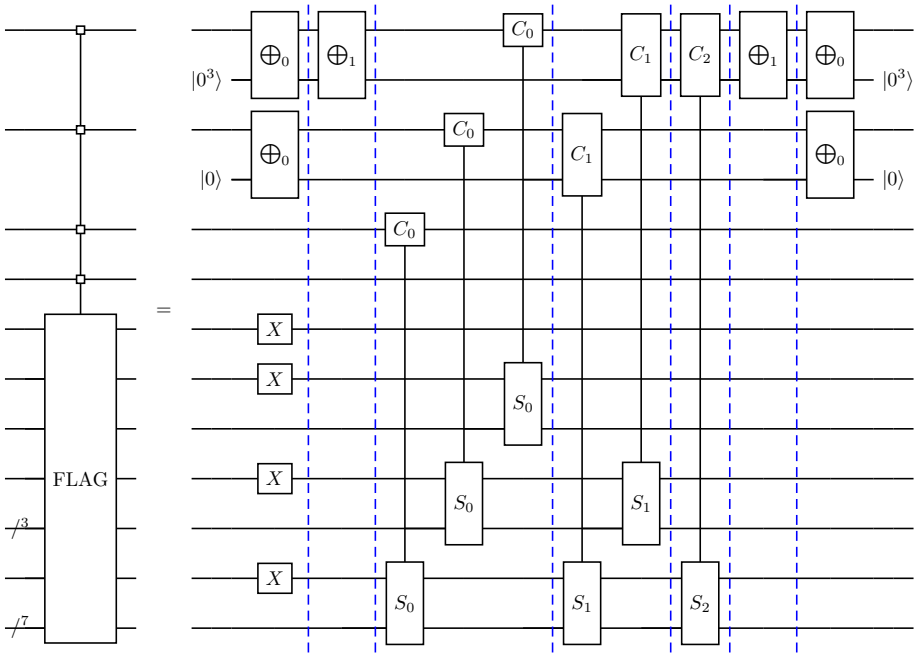

Figure 17: FLAG circuit for $m = 4$. All the gate sequences within each of the two blue dashed lines can be executed in parallel.

Algorithm 6 FLAG Operation

procedure FLAG SUBROUTINE(D, F, m)
▷ $D = D_0, \ldots, D_{m-1}$ is the data register, where D_j has size $\max(1, 2^{m-j-2})$
▷ $F = F_0, \ldots, F_{m-1}$ is the flag register where F_j has size 2^j
  for j **in** range(m) **do** ▷ All values of j performed in parallel
    X($F_{j,0}$)
  end for
  for i **in** range $(m-1)$ **do** ▷ All values of i performed in parallel
    COPY(D_i) ▷ Occupies maximum of $O(m)$ depth
  end for
  for i **in** range($m-1$) **do** ▷ Each value of i occupies $O(1)$ depth
    for q **in** range($m-i-1$) **do**
      CS$_i$(D_q, F_{q+1+i})
    end for
  end for
  for i **in** range $(m-1)$ **do** ▷ All values of i performed in parallel
    COPY†(D_i) ▷ Occupies maximum of $O(m)$ depth
  end for
end procedure

▷ Total D$_{\text{exact}}$: $O(m)$
▷ Total D$_{\text{approx}}$: $O(m)$
▷ Total SA$_{\text{exact}}$: $O(m2^m)$
▷ Total SA$_{\text{approx}}$: $O(m2^m)$

A.6 Copy Swap Operation

To simplify the circuit logic of the LOADF subroutines, we define another subroutine operation $\overline{\text{CopySwap}}$. To define $\overline{\text{CopySwap}}$, let $|k\rangle$ be an m-qubit computational basis state, written in binary as $k_{m-1}k_{m-2}\dots k_0$, so that k_0 represents the least significant bit. Let $|\xi\rangle$ be an arbitrary single-qubit state. The operation $\overline{\text{CopySwap}}$ enacts the isometry from an $m+1$ qubit state to a $2M-1$ qubit space

$$\overline{\text{CopySwap}}\left(|k\rangle|\xi\rangle\right) = |k_{m-1}\rangle^{\otimes 2^{m-1}} |k_{m-2}\rangle^{\otimes 2^{m-2}} \dots |k_1\rangle^{\otimes 2} |k_0\rangle |0^k\rangle |\xi\rangle |0^{M-k-1}\rangle \tag{35}$$

for any m-qubit computational basis state $|k\rangle$ and any arbitrary single-qubit state $|\xi\rangle$. Implementing $\overline{\text{CopySwap}}$ efficiently will involve a combination of copying layers and layers of parallel CSWAPs, depicted in Fig. 18 using the $\bigoplus_t$ and CS_t subroutines defined in App. A.1. Similar to previous sections, the goal of performing many $\bigoplus_t$ operations before the corresponding CS_t sequences at the earliest time is to ensure the CS_t sequences can be maximally parallelized under 1 step.

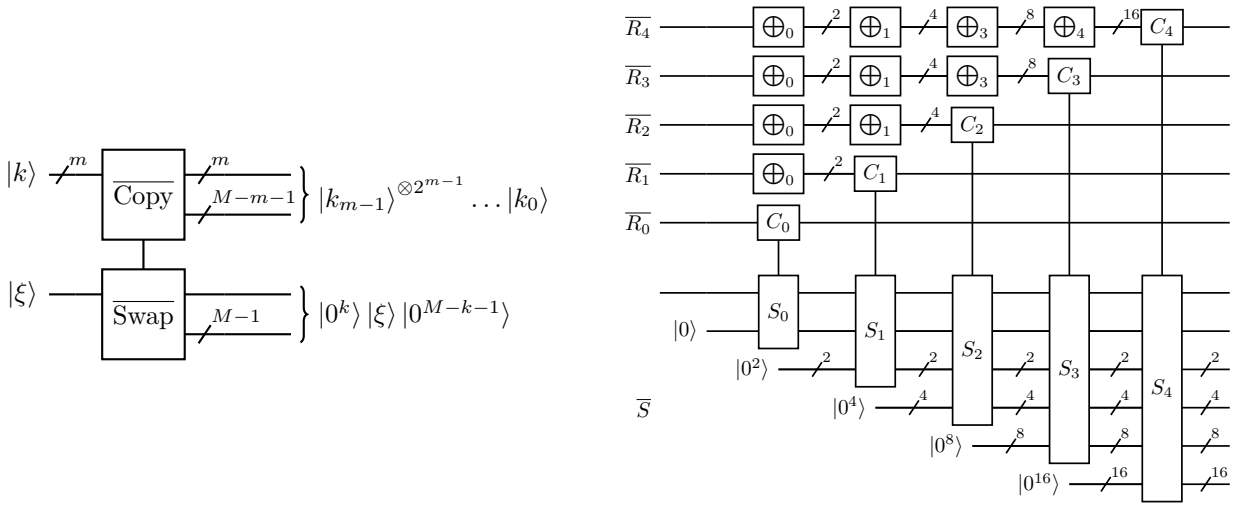

Figure 18: Left: action of $\overline{\text{CopySwap}}$ operation, which simultaneously copies m control bits set to $|k\rangle$ and moves a target register into the k^{th} position of a target register. The input is $m+1$ qubits and the output is $2M-1$ qubits, as several fresh ancillae are introduced during the protocol. Right: implementation of $\overline{\text{CopySwap}}$ operation using copying layers and swap operations for $m=5$. The total depth is m. At layer $t=0,\dots,m-1$, there are $(m+(2^t-1)(m-t)+2^{t+1})$ active qubits so the total spacetime allocation is $\sum_{t=0}^{m-1}(m+(2^t-1)(m-t)+2^{t+1}) = O(2^m)$.

Algorithm 7 Copy Swap Operation

procedure $\overline{\text{CopySwap}}$ SUBROUTINE($\overline{R}, \overline{S}, m$)
  ▷ m is the number of control bits
  ▷ $\overline{R} = \overline{R_0},\dots,\overline{R_{m-1}}$ is the control register, where $\overline{R_j}$ has size $\max(1, 2^{m-j-2})$
  ▷ $\overline{S}$ is the target register of size 2^m
  for i **in** range(m) **do** ▷ $O(m)$
    for j in range(1, m - i) **do** ▷ All values of j performed in parallel with the CS operation
      $\bigoplus_i(\overline{R_j})$
    end for
    $\text{CS}_i(\overline{R_i}, \overline{S})$
  end for
end procedure

▷ Total D_{exact}: $O(m)$
▷ Total D_{approx}: $O(m)$
▷ Total SA_{exact}: $O(m2^m)$
▷ Total $\text{SA}_{\text{approx}}$: $O(m2^m)$

A.7 LOADF

The LOADF circuit is perhaps the key to making the whole protocol work. It has two control registers, a data register of size m and a flag register of size $\tilde{B} := N/M - 1$. If the data register is set to $|k\rangle$, LOADF reads in the $\tilde{B}$ angle states needed to create the associated $(n-m)$-qubit state, but the angle associated with indices (s,p) is only read in if the flag in position (s,p) is set to 1. Mathematically, LOADF achieves the following operation:

$$\text{LOADF}\left(|k\rangle\,|0^{N/M-1}\rangle\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\rangle\right]\right) = |k\rangle\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\theta_{s,p}^{(k)}\rangle\right]\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\rangle\right] \tag{36}$$

One key feature of LOADF is that all of the single-qubit rotation gates occur in the same layer of the circuit. This allows it to achieve $O(\log(N)+\log(1/\epsilon))$ depth in the approximate setting. Another feature is that the $O(N)$ ancilla qubits are needed only very briefly to act as copied controls for doubly controlled rotation gates. As soon as the rotation gates are finished, the ancilla qubits can be rapidly deallocated, allowing the protocol to achieve $O(N)$ spacetime allocation. The protocol for LOADF is depicted in Fig. 19, with the corresponding pseudocode written in Algorithm 8.

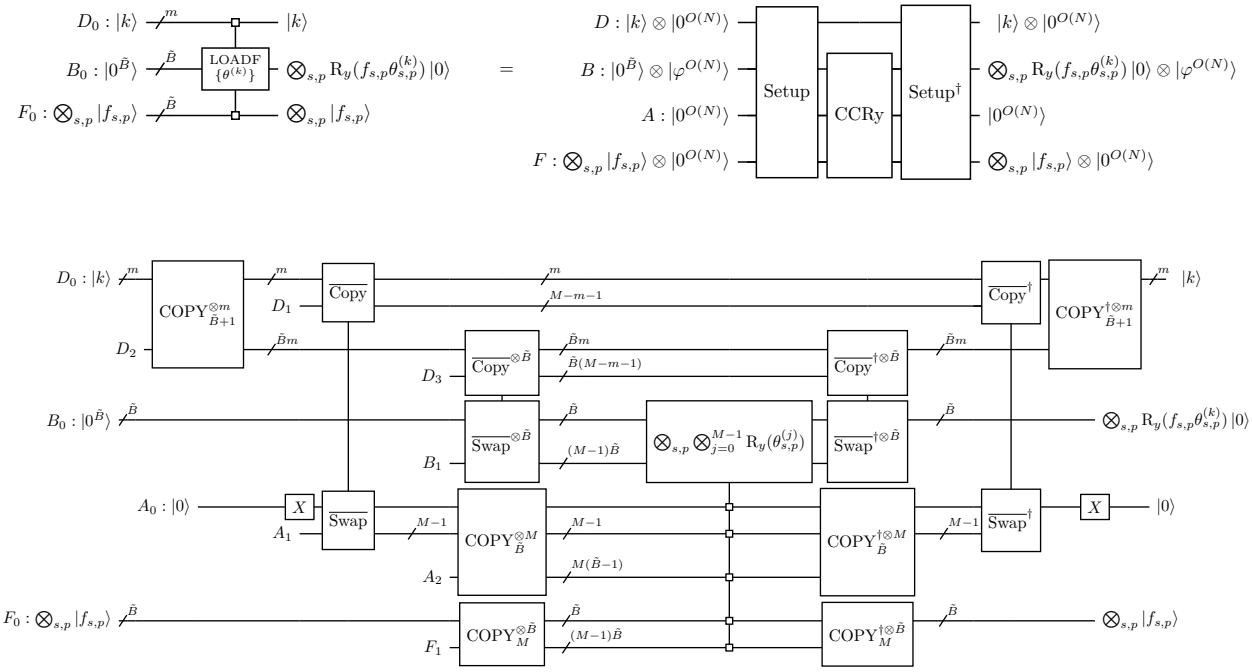

Figure 19: Top: Modular abstraction of LOADF. Setup is used to prepare the expanded address based on the address $|k\rangle$. Then CCRy gates pump in the angles using the expanded address in parallel. Lastly, Setup† uncompute all the expanded address. Bottom: Circuit implementing LOADF. The numerical value $\tilde{B}$ is defined as $\tilde{B} := N/M - 1$. The layer of rotation gates represents $M\tilde{B} = N - M$ parallel CCR_y gates by different angles $\theta_{s,p}^{(k)}$, with $k = 0,\ldots,M-1$, $s = 0,\ldots,n-m-1$, and $p = 0,\ldots,2^s-1$. For each CCR_y gate, one qubit from the final two registers (corresponding to the choice of (s,p)) and one qubit from the three registers above the final two registers (corresponding to the choice of k) acts as a control. Copying layers before the CCR_y sequence is necessary such that these can all be performed in parallel. After the CCR_y sequence, we perform $\tilde{B}$ $\overline{\mathrm{CopySwap}}^\dagger$ operations in parallel in order to load the $N-M$ angles into the buffer ancilla of size $\tilde{B}$. It can be verified that LOADF achieves $O(N)$ spacetime allocation: the only steps where $O(N)$ ancillae are active are the rotation gates (depth $O(1)$) and the uncopying step that follows. However, the uncopying step for $O(N)$ qubits was seen in Figs. 18 and 12 to occupy only $O(N)$ spacetime allocation. Unless labeled otherwise, all ancilla qubits begin and end with the $|0\rangle$ states. We also note that register B can be mostly allocated with dirty qubits $|\varphi^{O(N)}\rangle$ [16] instead of qubits in the $|0\rangle$ states.

Algorithm 8 LOADF Subroutine

procedure LOADF SUBROUTINE(D, B, F)
▷ This subroutine will also involve additional O(N) ancilla qubits. They do not need to be allocated in each ancilla group until individually called on. Once uncomputed, they can be deallocated (i.e., able to participate in a different computing task).
▷ D_0 is the address register of size m. D_1, D_2, D_3 contain $O(N)$ additional ancillae needed to complete all the parallelized CSWAP and CCRy gates.
▷ B_0 is the "buffer" ancilla register of size $\tilde{B}$ to hold the angles for SPF to pump into the data registers. B_1 contains $O(N)$ additional ancillae needed to complete all the parallelized CSWAP and CCRy gates.
▷ F_0 is the FLAG register of size $2^{n-m}-1$ (only required for LOADF†, optional for LOADF. Since the FLAG bits are all in the $|1\rangle$ states and thus guaranteed the control requirement for the following CCRy gates, we can effectively treat them as CRy gates only controlled on the A register.) F_1 contains $O(N)$ additional ancillae needed to complete all the parallelized CSWAP and CCRy gates.
▷ A is the register of size $M-N$ to hold the $M-N$ expanded angle address (Not shown in Fig. 8).
▷ For the purpose of illustration, we ignore the particular indices in the subroutine parameters. We provide exact circuit-level implementations in 6 with all the indices specified.
  Setup Subroutine
  for i in range($N-M$) **do** ▷ All i in parallel, to load $N-M$ angles into $N-M$ qubits
    CCRy(F, A, B, θ) ▷ $O(1)(O(\log(n/\epsilon)))$ depth
  end for
  Setup† Subroutine
end procedure
procedure Setup SUBROUTINE(D, B, A, F)
  X(A) ▷ $O(1)$ depth
  for i in range(m) **do** ▷ all i in parallel
    COPY$^\dagger_{\tilde{B}+1}(D)$ ▷ $O(n-m)$ depth
  end for
  $\overline{\text{CopySwap}}(D, A, m)$ ▷ $O(m)$ depth
▷ The following 3 for loops can be done in parallel
  for i in range($\tilde{B}$) **do** ▷ all i in parallel
    $\overline{\text{CopySwap}}(D, B, n-m)$ ▷ $O(n-m)$ depth
  end for
  for i in range(M) **do** ▷ all i in parallel, to prepare control bits in the CCRy layer's control on the A registers
    COPY$_{\tilde{B}}(A)$ ▷ $O(n-m)$ depth
  end for
  for i in range($\tilde{B}$) **do** ▷ all i in parallel, to prepare control bits in the CCRy layer's control on the F register
    COPY$_M(F)$ ▷ $O(m)$ depth
  end for
end procedure
▷ Total D_{exact}: $O(n)$
▷ Total D_{approx}: $O(n+\log(1/\epsilon))$
▷ Total SA_{exact}: $O(2^n)$
▷ Total $\text{SA}_{\text{approx}}$: $O(2^n \log(n/\epsilon))$

When compiling the CSP circuit into the {H, S, T, CNOT} gate set, a significant quantum resource consumption ($O(\log(n/\epsilon))$ depth and $O(2^n \log(n/\epsilon))$ spacetime allocation) is contributed by the parallelized CCRy gates. As illustrated in Fig. 19, a majority of the qubits in B that performs the parallelized Ry gates can be dirty (See proofing code for a state preparation task ($m=1$, $n=3$) where we use time-dependent gates before and after each segment to emulate the dirty qubits in the B_1 register), which would be of abundant supply in fault-tolerant algorithms such as LCU [9]. The ancilla qubits required to perform the Toffoli operations (i.e., A and F registers) can be immediately uncomputed using COPY† once the Toffoli gates are done if the ratio between $\log(1/\epsilon)$ and n is large, thus making the required spacetime allocation of clean qubits $O(N/M \log(\log(N)/\epsilon)+N)$ rather than $O(N \log(\log(N)/\epsilon))$. Future work can also aim to modify the circuit structure to allow, e.g., D, A, and/or F registers to be dirty.

When considering the number of clean qubits required to perform Ry gates in the context of the whole circuit (both SP and CSP circuits require parallelized Ry gates), we can see that we essentially need $O(M)$ clean qubits for the SP circuit and $O(N/M)$ clean qubits for the CSP circuit. Given enough supply of dirty qubits in the case when we need to reach high quantum state precisions using the $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set, we can let $M = \sqrt{N}$ (in other words, $m = n/2$, which also satisfies Eq. (17)), so the total clean qubit count will now be upper bounded to $O(\sqrt{N})$, thus resulting in $O(\sqrt{N}\log(\log(N)/\epsilon) + N)$ spacetime allocation for the clean qubits.

We illustrate the early ancilla free-up advantage over previous work in Fig. 20 and Fig. 21 using Quirk [48]. As mentioned at the end of Sec. 1, previous methods (e.g., Clader et al., shown in Fig. 20, see code) require $O(N)$ ancilla qubits to be entangled with the $O(n)$ data qubits for $O(n)$ depth, leading to $O(n2^n)$ spacetime allocation. This work (illustrated in Fig. 21, see code, and also on the right part of Fig. 1), only requires most of the $O(N)$ qubits to be allocated briefly.

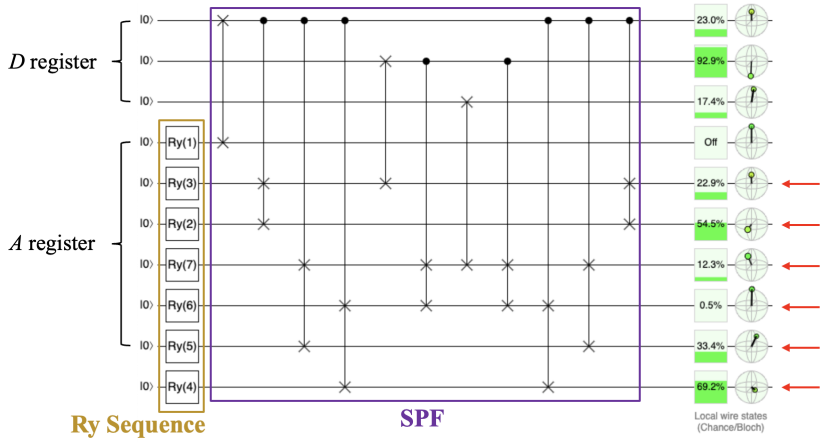

Figure 20: First part of Clader et al. [20] (aka U_{SP} without COPY). Note that all the ancilla qubits in the A register are left **entangled** with the data qubit in the D register (pointed by the red arrows). These $O(N)$ ancilla qubits are not returned to $|0\rangle$ until after the execution of the depth-$O(n)$ FLAG subroutine.

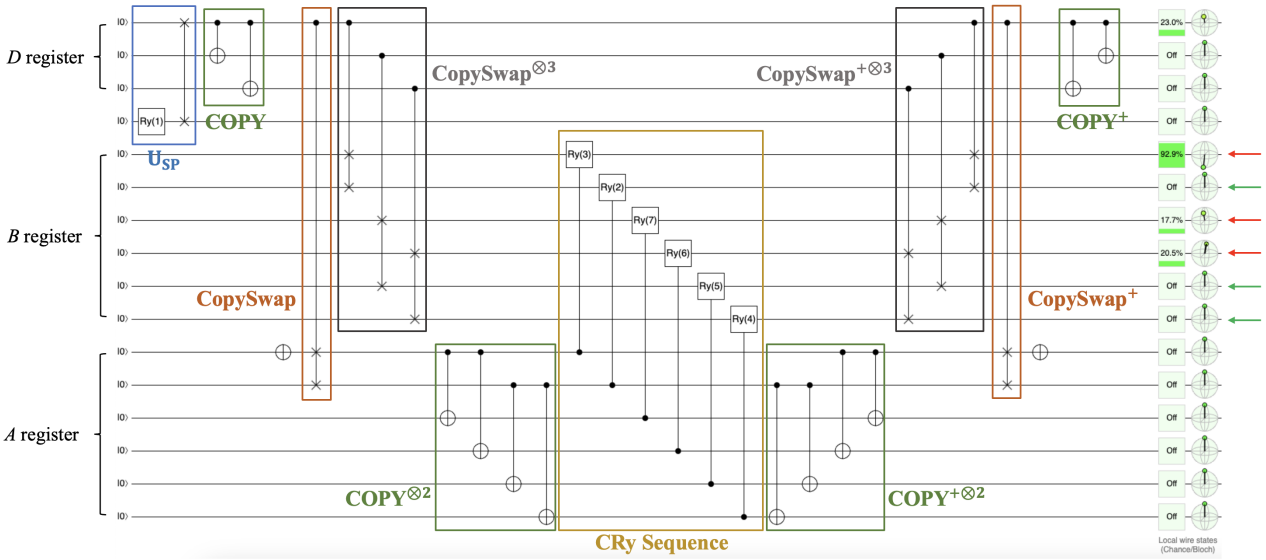

Figure 21: U_{SP} + LOADF part of the circuit developed in this work, on an example with $m = 1$ and $n = 3$. Note that most of the ancilla qubits in the B register are freed up (pointed by the green arrows). The remaining $O(\frac{M}{N})$ ancilla qubits (pointed by the red arrows) are freed up after the following SPF circuit that only takes $O(m)$ depth rather than $O(n)$ depth in the previous case. All other ancilla qubits in D, B, and A registers are freed up almost immediately after initialization. Note that as N grows larger and if the relation in Eq. (17) is satisfied, we can see a much larger ratio between the qubits labeled with the red arrows and qubits labeled with the green arrows, which shows the full advantage of this work's approach.

B Action of our protocol when the input state is not $|0^n\rangle$

Our state preparation procedure acts on n data qubits and uses a number $\ell = O(N)$ of ancilla qubits. It implements a unitary operation U on the $n + \ell$ qubits that sends $|0^n\rangle |0^\ell\rangle \mapsto |\psi\rangle |0^\ell\rangle$, that is, all the ancillae begin and end in $|0\rangle$. However, if the same unitary is performed on a state $|z\rangle |0^\ell\rangle$ for some $z \neq 0^n$ (or more generally, a state which is a superposition of all $z \neq 0^n$), it will no longer be the case that all ancillas are reset to $|0\rangle$. This results from the SPF circuit structure, described in Sec. A.4, which enacts rotations by swapping the n data qubits into the ancilla register. It exploits the fact that the data qubits are known to start in $|0\rangle$ in order to guarantee that the ancillae are left in the state $|0\rangle$.

This feature could be problematic in applications where the state preparation unitary is a part of a larger algorithm, and does not always act on the state $|0^n\rangle$. Is the procedure still useful in these other cases, and does it still achieve efficient spacetime allocation? Here we argue that the favorable properties of our procedure, in particular its optimal depth and spacetime allocation, extend to other common situations where state preparation appears, for example, implementing reflections and projections.

B.1 Implementing reflections about arbitrary states

A common use of state preparation within larger algorithms is to perform a reflection about a particular state $|\psi\rangle$, that is, the operation

$$R = I_n - 2 |\psi\rangle \langle\psi| \ , \tag{37}$$

where I_n denotes the identity operator on n qubits.

We now discuss how to implement this operator using U. First, we make a distinction between different ancilla registers in our state-preparation protocol. For the SP portion of the protocol, we have ancilla registers A and F, each of size $M - 1$, depicted in Fig. 7. For the CSP portion, we have an additional register B of size $N/M - 1$ (register F is the same as the one from the SP portion). The figures depict the registers A and B beginning and ending in $|0\rangle$, but this is only the case because some of the data registers have input $|0\rangle$. On the other hand, the F register has the property that it begins in $|0\rangle$ if and only if it ends in $|0\rangle$, regardless of the state of the other registers. Additionally, the implementation of SPF, FLAG, and LOADF introduce additional ancilla registers, as depicted in Figs. 15, 17, 19, but these ancilla registers are similar to the F register: they begin in $|0\rangle$ if and only if they end in $|0\rangle$, regardless of the state elsewhere in the circuit. Thus, we separate the ℓ ancillae into two groups, the group of $\ell' = M + N/M - 2$ ancillae in registers A and B, and the other $\ell'' = \ell - \ell'$ ancillae.

With this in mind, we can express

$$R \otimes |0^{\ell'}\rangle \otimes |0^{\ell''}\rangle = U \Big[I_n \otimes I_{\ell'} \otimes I_{\ell''} - 2 |0^n\rangle \langle 0^n| \otimes |0^{\ell'}\rangle \langle 0^{\ell'}| \otimes I_{\ell''} \Big] U^\dagger \Big[I_n \otimes |0^{\ell'}\rangle \otimes |0^{\ell''}\rangle \Big] . \tag{38}$$

Let us verify the above formula. When the input is $|\psi\rangle$, the action of $U^\dagger$ yields $|0^n\rangle |0^\ell\rangle$, a sign is applied, and application of U outputs the state $-|\psi\rangle |0^\ell\rangle$, as expected. When the input is $|\perp\rangle$ orthogonal to $|\psi\rangle$, application of $U^\dagger$ yields a state $|\perp'\rangle |0^{\ell''}\rangle$, where all we can guarantee is that $|\perp'\rangle$ is orthogonal to $|0^n\rangle |0^{\ell'}\rangle$. The reflection operation does not apply a sign, and subsequent application of U outputs $|\perp\rangle |0^\ell\rangle$, as expected. Crucially, this requires that we perform a reflection about both the n data qubits *and* the ℓ' ancillae that make up registers A and B all simultaneously being in $|0\rangle$ (but not the other ℓ'' ancillae as they are already guaranteed to be in $|0\rangle$). A reflection about t qubits being in the state $|0\rangle$ can be implemented in depth $O(\log(t))$ using $O(t)$ ancillae and $O(t)$ Toffoli gates by computing whether all qubits are set to $|0\rangle$ in a tree-like fashion, and applying a phase if the result is 1. Here $t = n + N/M + M - 2$, so the depth is $O(n)$. The spacetime allocation is upper bounded by the depth times the number of active qubits, i.e. $O(n \cdot \max(N/M, M))$. Since we choose M such that $\max(N/M, M) = O(N/n)$, this spacetime allocation is at most $O(N)$. In conclusion, our state preparation method can be used to perform reflections about arbitrary states on n qubits in depth $O(n)$ and spacetime allocation $O(N)$.

B.2 Implementing projections and block-encodings involving projections

Another operator where state-preparation is relevant is the projection onto the complement of an arbitrary state $|\psi\rangle$, that is

$$P = I - |\psi\rangle\langle\psi| \ . \tag{39}$$

This operator appears, for example, in the matrices that form the adiabatic path used in query-optimal quantum linear systems solvers [46]. In that context, Appendix F of Ref. [46] explains how a block-encoding of the relevant matrix can be formed given a state preparation unitary that maps $|0\rangle \mapsto |\psi\rangle$. This larger block-encoding involves constructing a block-encoding of the projector P, which is reduced to implementing a reflection about $|\psi\rangle$ (see Figs. 1–6 of Ref. [49], especially Fig. 4, for a quantum circuit interpretation of the discussion in Appendix F of Ref. [46]). As illustrated above, our method can perform reflections with optimal depth and spacetime allocation, and thus can also be applied in this application.

C Complex amplitudes

The constructions presented in the main text can prepare arbitrary states with real non-negative coefficients. Extending the construction to work for arbitrary coefficients is straightforward. The SP portion of the SP+CSP protocol is unchanged, as the state $|\phi\rangle$, as defined in Eq. (5) is insensitive to any phases in the vector $\mathbf{x}$ (all y_i are positive). On the other hand, the CSP portion of the protocol must be slightly modified. For each $k = 0, \ldots, M-1$, $j = 0, \ldots, N/M - 1$, define the phase

$$e^{i\varphi_j^{(k)}} = \frac{x_{j+kN/M}}{|x_{j+kN/M}|} \tag{40}$$

(if $x_{j+kN/M} = 0$, then the phase can be defined arbitrarily).

Next, we redefine the state $|\theta_{s,p}^{(k)}\rangle$:

$$|\theta_{s,p}^{(k)}\rangle = \begin{cases} \cos(\theta_{s,p}^{(k)}/2)\,|0\rangle + \sin(\theta_{s,p}^{(k)}/2)\,|1\rangle & \text{if } s < n-m-1 \\ e^{i\varphi_{2p}^{(k)}}\cos(\theta_{s,p}^{(k)}/2)\,|0\rangle + e^{i\varphi_{2p+1}^{(k)}}\sin(\theta_{s,p}^{(k)}/2)\,|1\rangle & \text{if } s = n-m-1 \end{cases} \tag{41}$$

To prepare $|\theta_{s,p}^{(k)}\rangle$ we need to apply a single-qubit rotation about the y-axis, and then, if $s = n-m-1$, we must apply a single qubit rotation about the z-axis, as well as a global phase (the global phase will be important when we add controls). To prepare a state with complex amplitudes, the only aspect of the protocol that has to change is the LOADF operation, which appears twice in the circuit for U_{CSP} in Fig. 8. As seen in Fig. 19, implementing LOADF involves a doubly controlled R_y gate. To account for complex amplitudes, we must augment this step with a doubly controlled R_z gate, as well as a doubly controlled global phase (which is equivalent to a singly-controlled R_z gate). These additions lead to at most a constant factor increase in the depth and spacetime allocation. In the approximate $\{\mathrm{H}, \mathrm{S}, \mathrm{T}, \mathrm{CNOT}\}$ gate set, all rotations need only be synthesized to error $O(\epsilon/n)$ to achieve overall error ϵ in the state preparation protocol, requiring only $O(\log(n/\epsilon))$ gates per rotation. None of the complexity statements in the paper are impacted.