Succinct Fermion Data Structures

Joseph Carolan
University of Maryland, College Park, MD, USA

Luke Schaeffer
University of Waterloo, Canada

Abstract

Simulating fermionic systems on a quantum computer requires representing fermionic states using qubits. The complexity of many simulation algorithms depends on the complexity of implementing rotations generated by fermionic creation-annihilation operators, and the space depends on the number of qubits used. While standard fermion encodings like Jordan-Wigner are space optimal for arbitrary fermionic systems, physical symmetries like particle conservation can reduce the number of physical configurations, allowing improved space complexity. Such space saving is only feasible if the gate overhead is small, suggesting a (quantum) data structures problem, wherein one would like to minimize space used to represent a fermionic state, while still enabling efficient rotations.

We define a structure which naturally captures mappings from fermions to systems of qubits. We then instantiate it in two ways, giving rise to two new second-quantized fermion encodings of F fermions in M modes. An information theoretic minimum of $\mathcal{I} := \lceil \log_2 \binom{M}{F} \rceil$ qubits is required for such systems, a bound we nearly match over the entire parameter regime.

1. Our first construction uses $\mathcal{I} + o(\mathcal{I})$ qubits when $F = o(M)$, and allows rotations generated by creation-annihilation operators in $O(\mathcal{I})$ gates and $O(\log M \log\log M)$ depth.
2. Our second construction uses $\mathcal{I} + O(1)$ qubits when $F = \Theta(M)$, and allows rotations generated by creation-annihilation operators in $O(\mathcal{I}^3)$ gates.

In relation to comparable prior work, the first represents a polynomial improvement in both space and gate complexity (against Kirby et al. 2022), and the second represents an exponential improvement in gate complexity at the cost of only a constant number of additional qubits (against Harrison et al. or Shee et al. 2022), in the described parameter regimes.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Data compression

Keywords and phrases quantum computing, data structures, fermion encodings

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.32

Related Version *Full Version*: `https://arxiv.org/abs/2410.04015`

Funding *Joseph Carolan*: supported by the US Department of Energy grant no. DESC0020264.

Acknowledgements The authors thank James Watson for helpful conversation about fermion encodings, and Andrew Childs for providing feedback on an early draft of this manuscript. LS thanks the Joint Center for Quantum Information and Computer Science (QuICS) where a large portion of this research occurred.

1 Introduction

The simulation of many interacting fermions is a promising application of quantum computers. Such systems quickly become difficult to simulate classically, with accurate simulation being computationally intractable for complex systems [28]. A quantum computer can perform highly accurate simulations with only polynomial resources for local systems, a wide and powerful class beyond what can be simulated classically.

Many quantum algorithms for fermionic simulation require a mapping from fermionic Fock states and creation/annihilation operators to qubit states and operators. Two of the most well known such mappings are the ones due to Jordan-Wigner [17] and Bravyi-Kitaev [7].

Table 1 Comparison to prior space efficient second-quantized encodings for F fermion, M mode systems, adapted from [19]. Gates refers to the complexity of implementing rotations generated by creation-annihilation operators. Here $\mathcal{I} := \left\lceil \log \binom{M}{F} \right\rceil$, which satisfies $\mathcal{I} = \Theta\left(F \log\left(\frac{M}{F}\right)\right)$.

Encoding	Qubits	Gates
Regime $F = o(M)$ and $F = \omega(1)$.		
Optimal Degree[19]	$\Omega(\mathcal{I}^2 \log^2 M)$	$\Omega(\mathcal{I}^2 \log^3 M)$
Qubit Tapering/Segment[6, 25, 24]	$M - o(M)$	$\Omega(F^2)$
Permutation Basis[14]	$\mathcal{I}$	$\Omega(M^2 2^{\mathcal{I}})$
Theorem 1	$\mathcal{I} + o(\mathcal{I})$	$O(\mathcal{I})$
Regime $F = \Theta(M)$		
Qubit Tapering/Segment[6, 25, 24]	$M - O(1)$	$\Omega(\mathcal{I}^2)$
Permutation Basis[14]	$\mathcal{I}$	$\Omega(\mathcal{I}^2 2^{\mathcal{I}})$
Theorem 2	$\mathcal{I} + O(1)$	$O(\mathcal{I}^3)$
General regime		
Theorem 1	$\mathcal{I} + O(F)$	$O(\mathcal{I})$
Theorem 2	$\mathcal{I} + O(\log(M/F))$	$O(MF\mathcal{I})$

These require M qubits to represent a fermion system with M modes, which is prohibitive in some cases. In particular, for particle preserving systems where the number of fermions F is much smaller than M, these encodings have a large amount of redundancy. In this case, the information theoretic lower bound of

$$\mathcal{I} := \left\lceil \log \binom{M}{F} \right\rceil \tag{1}$$

qubits is potentially much smaller than M.

Encodings which take advantage of this fact are relevant whenever the number of fermions is small compared to the number of modes, for instance in quantum chemistry simulations where the number of electrons (F) is fixed by a physical system yet the orbital basis size (M) should be as large as possible to correctly resolve the continuum behaviour. The discretization error in such systems scales like $1/M$ for reasonable bases [26], so large M is required for high accuracy – this is especially important when using basis sets that have not been classically optimized, such as the plane wave basis. This motivates developing simulation algorithms with as mild a dependence on M as possible in both space and gate usage, and a key component of many simulation algorithms is the qubit-to-fermion mapping used. Utilizing as few qubits as possible is important both for near term and fault tolerant devices, due to their limited size and the expense of qubits. However, space savings that incur an exponential gate overhead are not scalable. We therefore seek to minimize space complexity while still allowing relevant operations to be performed efficiently.

1.1 Prior work

There has been a line of work on second-quantized fermion encodings for saving space when the number of fermions is fixed. Bravyi et al. give a scheme for utilizing symmetries in particle preserving Hamiltonians to remove degrees of freedom, using parity symmetries to achieve $M - O(1)$ qubits generically and LDPC codes for few fermion systems to achieve $M - O(M/F)$ qubits [6], but with gate overhead $O(M^3)$ in the worst case[1]. Steudtner and Wehner give a

[1] Gate complexity here refers to the complexity of performing a single $\exp(-i\theta K)$, for some angle θ and number-preserving product of $O(1)$ many creation/annihilation operators K. For our discussions, we will not consider geometric locality.

scheme based on segmenting the space of fermions which again achieves $M - O(M/F)$ qubits, though allows more efficient $O(F^2)$ gate overhead [25, 24]. The same work also proposed a binary addressing code which shares some similarities with the encodings described here, but did not provide circuits for fermion operations nor bound the resource costs in the general case. The work of Babbush et al. [2] describes a way to store and manipulate few fermion systems using a sparse oracle, but do not provide explicit circuits for particle preserving rotations generated by creation-annihilation operators.

The work of Kirby et al. [19] describes a second quantized fermion encoding achieving $O(F^2 \log^4 M)$ space complexity and $O(F^2 \log^5 M)$ gate complexity, subject to a certain conjecture about Hermite interpolation. Additionally, the work of Harrison et al. [14] and Shee et al. [23] give methods for achieving exactly $\mathcal{I}$ qubit usage in second quantization, but at the cost of an exponential $M^{O(F)}$ gate complexity.

There are also first-quantized representations of fermion systems [26, 18, 1], where a list of occupied fermion modes is anti-symmetrized. This can save significant space when F is much smaller than M. However, such encodings allow for a different class of operations[2] to be performed efficiently as compared to second quantized encodings, making circuit complexity results within these two paradigms not directly comparable. Other works consider utilizing more than M qubits and restricting to an entangled subspace, allowing certain sets of k-local fermion operations to be extremely efficient (see e.g. [29, 30, 11, 7]). These encodings are most relevant for geometrically local systems, and in fact increase space beyond the standard Jordan-Wigner encoding – we therefore omit them from comparisons.

1.2 Contributions

We describe a quantum data structure which naturally captures the problem of encoding (second quantized) fermion systems in qubits. This structure represents bitstrings of length M and Hamming weight F, which naturally correspond to fermionic Fock states. We consider the (quantum circuit) gate/depth complexity of a sgn-rank ("sign rank") operator, which applies a phase conditioned on how many ones exist in some prefix of the bitstring, as well as a bit-flip operator which flips one of the bits. These operators allow us to straightforwardly express the fermionic operations which are relevant to quantum simulation, as we show in Section 3.

Within this framework, the Jordan-Wigner encoding can be seen as a data structure which encodes bitstring $b \in \{0,1\}^M$ by the quantum state $|b\rangle$, e.g. the trivial encoding. The sign rank operation is then simply a contiguous string of Z operators on some prefix, and the bit flip operator is a single X operator. Our encodings employ a different representation of bitstrings which is more efficient when F is small, requiring different quantum circuits.

1.2.1 Succinct structure

We first give a second quantized encoding for number preserving fermion systems which is nearly optimal in space usage, yet allows fermionic rotations with linear complexity and low depth. Our encoding uses sublinear redundancy, i.e. it is within a factor $1 + o(1)$ of optimal space usage (this is the meaning of "succinct" in the context of data structures) whenever $F = o(M)$. In particular, we prove the following theorem.

[2] In particular, first-quantized operations. See [26] for examples of usage of these encodings in quantum simulation.

▶ **Theorem 1** (Informal). *A system of F fermions in M modes can be represented by $\mathcal{I}+O(F)$ qubits such that particle preserving fermionic rotations*[3] *have circuits of complexity $O(\mathcal{I})$ and depth $O(\log M \log\log M)$.*

A summary of comparable prior work is shown in Table 1; notably, all previous second-quantized encodings either incur a polynomial overhead in space complexity[4], or the gate complexity of fermionic rotations is exponential in the number of qubits. We build up to this main result by presenting intermediary encodings achieving some, but not all of the aforementioned scalings. These encodings may be of independent interest due to their simplicity.

1.2.2 Implicit structure

We also give a construction which uses essentially the exact information-theoretic space minimum, except for an $O(1)$ number of ancilla (at constant filling). This construction achieves $\mathsf{poly}(M)$ gate complexity. The relevant regime for this encoding is constant filling, e.g. $F = M/3$ or more generally $F = \Theta(M)$. In this regime, the prior space complexity of $\mathcal{I} + O(F)$ may be a significant overhead, whereas when $F = o(M)$ the additive $O(F)$ term is asymptotically negligible. The information theoretic limit in the $F = \Theta(M)$ regime goes like $\mathcal{I} = \Theta(F)$, meaning there can be significant space to save if F is any constant fraction below $M/2$.

▶ **Theorem 2** (Informal). *A system of F fermions in M modes can be represented by $\mathcal{I} + O(\log(M/F))$ qubits such that particle preserving fermionic rotations have circuits of complexity $O(MF\mathcal{I})$.*

The most comparable prior work is that of Harrison et al. [14] and Shee et al. [23], which both achieve exactly $\mathcal{I}$ qubit complexity. However, these require an $\Omega\left(\binom{M}{F}\mathsf{poly}(M)\right)$ (e.g. exponential in M at constant filling) circuit complexity overhead for performing rotations generated by creation-annihilation operators, in the worst case. Our encoding improves exponentially on this gate complexity, achieving $\mathsf{poly}(M)$ in this regime. We do this at the cost of $O(1)$ ancilla qubits.

1.3 Technical overview

We begin in Section 3 by developing a data structure which naturally captures fermion to qubit mappings of number preserving systems. This definition is not fully general, but provides a framework for thinking about such mappings. We then instantiate this data structure in two ways.

1.3.1 First result: a succinct structure

To build up to the first main result, we begin by developing a fermion data structure in Section 4 based on a straightforward idea. Instead of storing the explicit string of ones and zeros, store a sorted list of pointers to the $\leq F$ positions containing a one. This approach builds

[3] By fermionic rotation, we mean the unitary $\exp(-i\theta K)$ for angle θ, where K is a number-preserving product of creation/annihilation operators, e.g. $K = a_j^\dagger a_k$ for some $j, k \in [M]$. We require that K is $O(1)$ local in the sense that it is the product of $O(1)$ creation/annihilation operators (but not geometrically local).

[4] i.e. have asymptotic space usage $\Omega(\mathcal{I}^c)$ for some $c > 1$ when $F \ll M$.

off ideas presented in prior fermion encodings [2, 25] as well as work compressing quantum states [8, 13], though we present these ideas in a way which will facilitate understanding our full encoding. Observe that, in contrast to first quantized representations [26] (which also store a list of pointers to occupied positions), we will not antisymmetrize the state. This enables us to efficiently implement sgn-rank and bit-flip queries (analogous to second-quantized rather than first-quantized operations), which we provide in Section 4.2. In Section 4.3 we describe a procedure using buffer registers that reduces the depth in this construction to logarithmic.

From this simple starting point, our next fermion data structure in Section 5 reduces the space requirements using combinatorial ideas, without affecting gate complexity. We accomplish this by splitting the most and least significant bits of each register, and using different representations for each. The most significant bits have a large amount of redundancy from being non-increasing, which we avoid by storing them using the stars and bars method. We give circuits for combining information between these two representations to efficiently implement the required operations.

We combine the two previous ideas with a succinct data structure that enables low-depth access to the most significant bits to show Theorem 1, with details given in the full version.

The key technical idea in this construction is a tree data structure used to store prefix sums related to the most significant bits. This prefix sum tree allows retrieving information about any specific entry in low depth, but requires less total space than a sorted list of numbers. This data structure therefore allows log depth fermionic rotations while being near-optimal in space usage, our first main result.

1.3.2 Second result: an implicit structure

Using a different approach, detailed in the full version, our second data structure achieves $\mathcal{I} + O(1)$ qubits and $\mathsf{poly}(M)$ gate complexity when $F = \Theta(M)$. We represent a bit string b (which itself represents a Fock state) by storing b's rank among feasible strings, where we determine rank by the number of bit-strings which are lexicographically before b or have a smaller Hamming weight than b. This encoding uses $\mathcal{I} + O(1)$ many qubits, but it is unclear how to interpret the encoded string to perform efficient operations. This obscurity is, essentially, the reason for the exponential gate overhead in prior works achieving this level of space efficiency.

We show how to perform both bit-flip and sign-rank efficiently on the first (unencoded) position, intuitively because the first and most-significant bit determines whether a given string has rank at most $\binom{M-1}{F-1}$ (first bit a 0) or rank above $\binom{M-1}{F-1}$ (first bit a 1) among strings of a given Hamming weight. This means that we can extract the first bit by comparing the encoded label against a fixed value, which can be done efficiently and with a small number of ancilla.

We then transform the ordering to one in which bit-flip and sign-rank can be performed efficiently on the second bit, and so on for each position. This is based on a construction of a certain set of orderings, which we denote $<_j$ for $j \in [0, ..., M]$, of the labels. These orderings have the property that, given just the rank of a configuration over order $<_j$, operations on the j-th encoded bit can be performed efficiently ($O(F\mathcal{I})$ gates) and with few ($O(1)$) ancillas on the label string. Further, we show how to transform a label under the order $<_j$ to a label under either $<_{j+1}$ or $<_{j-1}$ in the same complexity. By cycling over the M possible orderings, we can perform relevant operations with $O(MF\mathcal{I})$ gates and $O(1)$ ancillas.

2 Preliminaries

2.1 Fermions

A system of fermions with M modes can be defined by the algebra of creation ($a_i^\dagger$ for $i \in [M]$) and annihilation (a_i for $i \in [M]$) operators – this formalism is referred to as second quantization. These operators are determined by the anti-commutation relations

$$\{a_i^\dagger, a_j\} = \delta_{ij}, \qquad \{a_i^\dagger, a_j^\dagger\} = 0, \qquad \{a_i, a_j\} = 0. \tag{2}$$

Let $|00...0\rangle_f$ denote the vacuum state, the unique state which is not annihilated by any of the $a_i^\dagger$. We will primarily work with the Fock states, which are of the form

$$\begin{aligned} |\psi^{\mathbf{b}}\rangle_f :=& (a_1^\dagger)^{b_1}(a_2^\dagger)^{b_2}...(a_M^\dagger)^{b_M}\,|00...0\rangle_f \qquad \text{(for all } \mathbf{b} \in \{0,1\}^M\text{)} \\ =& |b_1, b_2, ..., b_M\rangle_f\,. \end{aligned} \tag{3}$$

We denote the span of these states $\mathcal{H}^{(fock)}$. Note that the M-mode Fock states are in natural correspondence with M-bit strings – this observation underlies the famous Jordan Wigner encoding [17], and will similarly underlie our data structures. It will be convenient for us to work in the Majorana basis determined by γ_j for $j \in [2m]$. These operators are defined as follows:

$$\gamma_{2j-1} = \frac{a_j^\dagger + a_j}{2}, \qquad \gamma_{2j} = \frac{i(a_j^\dagger - a_j)}{2}, \qquad \text{such that: } \{\gamma_j, \gamma_k\} = \delta_{jk}. \tag{4}$$

The action of Majorana operators on the Fock states can now be written as follows, where $\neg b$ is the negation of bit b,

$$\begin{aligned} \gamma_{2j-1}\,|b_1...b_j...b_M\rangle_f &= \left(\prod_{n=0}^{j-1}(-1)^{b_n}\right)|b_1...(\neg b_j)...b_M\rangle_f\,, \\ \gamma_{2j}\,|b_1...b_j...b_M\rangle_f &= i\cdot\left(\prod_{n=0}^{j}(-1)^{b_n}\right)|b_1...(\neg b_j)...b_M\rangle_f\,. \end{aligned} \tag{5}$$

In particular, these operators flip the occupation of a certain mode and apply a phase depending on the occupation of all preceding modes.

2.2 Fermion to Qubit Mappings

A fermion to qubit mapping can be defined by a linear mapping $\mathcal{E}_s$ from Fock states to qubit states, as well as a mapping $\mathcal{E}_o$ from Majorana operators to qubit operators. The qubit states/operators should be isomorphic to Fock states and Majorana operators, i.e. satisfy Equations 4, 5. Though some encodings do not fit into this paradigm (discussed in Section 1.1), this definition suffices for the encodings relevant to this paper. In particular, to define a mapping of an M mode system it suffices to construct $2M$ mutually anti-commuting qubit operators which each square to the identity.

We note that it is not always necessary to encode every possible Fock state, e.g. when simulating a system that does not explore every state. Particularly relevant for us will be particle preserving systems, which live in the subspace of Fock states with exactly F many fermions.

A well known example of a fermion-to-qubit mapping is the Jordan-Wigner encoding [17], which defines M-qubit operators as follows (where P_i denotes a Pauli P on the i-th qubit, acting trivially on the rest)

$$\mathcal{E}_{JW}(\gamma_{2i-1}) = \left(\prod_{j=1}^{i-1} Z_j\right) \otimes X_i, \qquad \mathcal{E}_{JW}(\gamma_{2i}) = \left(\prod_{j=1}^{i-1} Z_j\right) \otimes Y_i. \tag{6}$$

The mapped qubit operators in this encoding act on M qubits, which means M qubits are used to store fermionic states. Additionally, the circuit complexity of the mapped operators is $\Omega(M)$, as some mapped operators act non-trivially on all qubits.

2.3 Simulating Fermions

The main use case for fermion to qubit mappings is in the simulation of interacting fermions. In many such applications the dynamics are governed by a Hamiltonian H which is both k-local and term-wise number preserving (i.e. each term commutes with $\sum_i a_i^\dagger a_i$). If we wish to simulate such a system with M modes, the most general Hamiltonian is

$$H = \sum_{l\leq k; l \text{ even}; i_1 i_2 \ldots i_l \in [M]} h_{i_1 i_2 \ldots i_l} a_{i_1}^\dagger a_{i_2} \ldots a_{i_{l-1}}^\dagger a_{i_l} + h.c. \tag{7}$$

where the indices run over the M modes and $h.c.$ denotes the hermitian conjugate of the preceding term. This type of system is ubiquitous in quantum chemistry and physics. To simulate a fermionic system, one requires a mapping from fermion states/operators to qubit states/operators, as discussed in Section 2.2. Almost all second quantized algorithms for approximating evolution $\exp(-iHt)$ involve performing rotations generated by terms in H [9, 10, 4, 3, 20, 5, 21], so it is better for these rotations to require few gates and act on few qubits. Let V be a product of k Majorana operators, where k is even. Our encodings give efficient circuits for rotations of the form $\exp(-i\theta V)$ (k a multiple of 4) or $\exp(-\theta V)$ (k not a multiple of 4)[5]. This is sufficient to implement any rotation generated by the product of $O(1)$ creation/annihilation operators and its hermitian conjugate. For an illustrative example, consider a hopping term $a_j^\dagger a_k + h.c.$ for $j \neq k$,

$$\begin{aligned}\exp\Big(-i\theta(a_j^\dagger a_k + a_k^\dagger a_j)\Big) &= \exp(2\theta(\gamma_{2j-1}\gamma_{2k} - \gamma_{2j}\gamma_{2k-1})) \\ &= \exp(2\theta\gamma_{2j-1}\gamma_{2k}) \exp(-2\theta\gamma_{2j}\gamma_{2k-1}),\end{aligned}$$

from which it is clear that it suffices to implement two Majorana rotations. Note that although each Majorana operator itself need not preserve particle number, it changes the occupation number by at most k (in this case at most two). Further, the whole operator does preserve particle number. Therefore, so long as our encoding has enough "space" to fit the intermediate states, we will remain in the proper subspace after implementing the full operator $\exp\Big(-i\theta(a_j^\dagger a_k + a_k^\dagger a_j)\Big)$. We refer the reader to the full version for a more general discussion of this procedure.

[5] Actually, in most places we give circuits for the operator V up to a global phase. In the full version, we discuss how to generically implement the aforementioned rotations in the same complexity. Further, whether k is a multiple of 4 dictates whether $\exp(-i\theta V)$ or $\exp(-\theta V)$ is unitary; we implement the unitary one.

3 Fermion Data Structures

In this section we introduce an alternative perspective of fermion encodings, in a way which makes the connection to data structures explicit. This perspective will facilitate the development of efficient fermion-to-qubit mappings later on.

3.1 Definitions

We first define two useful combinatorial operations, sgn-rank and bit-flip. These are analogous to rank and bit flip queries respectively, which are well-studied in succinct classical data structures for bit-vectors [15, 22][6]. These operators act on a bit string $\mathbf{b} = (b_1, ..., b_M) \in \{0,1\}^M$ and depend on an index $j \in [M]$, and are combinatorial operations with no direct relation to fermions. However, they both show up naturally when writing down the action of fermionic creation-annihilation operators on Fock states: one can notice the similarities between Definitions 3, 4 and the action of Majorana operators in Equation 5.

▶ **Definition 3.** *The operator* sgn-rank *("sign rank") of an index $j \in [M]$ and bit string $\mathbf{b} \in \{0,1\}^M$ is the operator which returns the parity of the first j bits of $\mathbf{b}$ (represented as $+1$ for even, -1 for odd). In particular,*

$$\mathsf{sgn\text{-}rank}(j, \mathbf{b}) := \prod_{n=0}^{j} (-1)^{b_n}.$$

▶ **Definition 4.** *The* bit-flip *operator of an index $j \in [M]$ and bit string $\mathbf{b} \in \{0,1\}^M$ flips the j-th bit of $\mathbf{b}$. In particular, we have the following, where $\neg$ is logical negation,*

$$\mathsf{bit\text{-}flip}(j, \mathbf{b}) := (b_1, ..., (\neg b_j), ..., b_M).$$

With these two operations in hand, we can now describe the relevant type of (quantum) data structure for encoding fermionic states. Intuitively, we would like to represent a bit string $\mathbf{b} \in \{0,1\}^M$ in such a way that both sgn-rank and bit-flip queries can be implemented efficiently. Furthermore, in the settings we will care about we will be promised that the Hamming weight of $\mathbf{b}$ (the number of 1's) will never exceed $F + k$ or subceed $F - k$, for some $F < M$ and constant $k = O(1)$ (see the full version for more discussion). We use the notation $\mathcal{H}^{(fock)}_{M,F,k}$ to denote the Fock space of an M-mode system of fermions, restricted to states with occupation between $F - k$ and $F + k$: we call this range the capacity. We use the notation $\mathcal{H}^{(qubit)}_n$ to denote the Hilbert space of n qubits.

▶ **Definition 5.** *A* fermion data structure *of size M and capacity F, k is a qubit representation of Fock states with M modes and occupation at most $F + k$ and at least $F - k$, i.e. a linear and invertible mapping $\mathcal{E}_s : \mathcal{H}^{(fock)}_{M,F,k} \to \mathcal{H}^{(qubit)}_n$ that maps Fock states to computational basis states. We denote $\mathcal{E}_s(|\mathbf{b}\rangle_f)$ as $|\mathbf{b}\rangle$. We further require two n-qubit quantum circuit families F_j and R_j, answering* bit-flip *and* sgn-rank *queries respectively. Formally, we require*

$$F_j |\mathbf{b}\rangle := |\mathsf{bit\text{-}flip}(j, \mathbf{b})\rangle \qquad (\textit{if } \mathsf{bit\text{-}flip}(j, \mathbf{b}) \textit{ has } F - k \leq \mathsf{HW} \leq F + k), \tag{8}$$

$$R_j |\mathbf{b}\rangle := \mathsf{sgn\text{-}rank}(j, \mathbf{b}) |\mathbf{b}\rangle . \tag{9}$$

[6] Note that the differing models of complexity (RAM programs/cell probe versus quantum circuits) prevent most of these classical results from being directly applicable to our problems.

The gate/depth complexity of the above structure is the max gate count/depth, respectively, of the F_j, R_j over all $j \in [M]$. The space complexity is n – note that ancillas used in F_j, R_j are counted in this complexity, as they are circuits acting on exactly n qubits.

We remark that we consider quantum circuit complexity as the relevant cost measure, which prevents us from using many standard data structures results. Classical results often use the cell probe model [12], which is based on a RAM machine that allows random access to data. This model may not capture complexity in a real world quantum computer [16]. This necessitates new data structures and new ideas, which will be the focus of this paper.

3.2 Properties of fermion data structures

We first observe a subtlety in Definition 5: the operation of bit-flip is only required to be "correct" so long as the capacity is not exceeded. When one is interested in simulating a k-local Hamiltonian on a system with exactly F Fermions, then it often suffices to instantiate a Fermion data structure with capacity F, k[7]. In particular, it is straightforward to show that such a data structure can be used to instantiate a Trotterization scheme, or any simulation algorithm which relies on applying k-local particle-preserving rotations generated by products of k creation/annihilation operators. Intuitively, so long as we can "fit" the intermediate states necessary while performing these rotations, we will be able to perform any arbitrary sequence of rotations.

▶ **Proposition 6.** *Let $|\psi\rangle_f$ be a fermionic Fock state of F fermions in M modes, and C_f be a sequence of k-local fermionic rotations which preserve particle number, followed by a measurement in the Fock basis. If there is a fermion data structure of size M and capacity $F-k, ..., F+k$, using space n and gate complexity m, then there is a qubit state $|\psi\rangle$ on n qubits and a quantum circuit C of size $O(|C_f| \cdot m)$ such that measuring $C|\psi\rangle$ in the computational basis samples from the same distribution as $C_f|\psi\rangle_f$ (up to permuting bit-string labels).*

Proof. The initial state $|\psi\rangle$ is simply the encoding $\mathcal{E}_s(|\psi\rangle_f)$. To construct the circuit C, we will replace every gate of C_f (which are of the form $g_f = \exp(i\theta V)$ for k-local, particle preserving V) with the corresponding encoded circuit for performing the rotation, as detailed in Section 2.3. Let us call this encoded circuit g. Letting $|\phi\rangle$ be some superposition of Fock states with F fermions, then with a data structure $\mathcal{E}_s$ of capacity $F-k, ..., F+k$ we have the guarantee that

$$g\mathcal{E}_s(|\phi\rangle) = \mathcal{E}_s(g_f|\phi\rangle),$$

as g_f corresponds to a sequence of sgn-rank and bit-flip operations satisfying the necessary promises. Noting that $g_f|\phi\rangle$ will also have F fermions by the hypothesis that g_f preserves particle number, we can similarly perform this transformation for every following gate. Finally, by the invertibility of $\mathcal{E}_s$ we will obtain the same final output distribution by measuring in the computational basis, up to decoding the bit-strings (i.e. some permutation on the labels). ◀

Given the above discussion, the curious reader may note that our definition of fermionic data structures encodes all states of Hamming weight between $F-k$ and$F+k$ for some constant k. This is redundant in that $F+k > F$, and also that there are more states having Hamming weight between $F-k$ and $F+k$ than there are of Hamming weight exactly equal

[7] Note that this does incur some space overhead, but it is only an additive $O(1)$ (if $F = \Theta(M)$) or $O(\log M)$ (if $F = O(M/\log M)$)

to $F + k$. However, both of these redundancies result in negligible additive space overheads (either $O(1)$ or $O(\log M)$ depending on the regime). The conclusions we draw about space efficiency includes these overheads.

We remark that our encoding can also apply to some post-Trotter methods, e.g. certain linear combination of unitaries algorithms. We also remark for clarity that the encodings leading up to Theorem 1 will allow encoding every Fock state of occupation at most $F + k$ (though those leading up to Theorem 2 will only encode occupations $F - k$ through $F + k$).

4 Sorted List Fermion Data Structure

In this section, we describe a fermion data structure (Definition 5) based on storing a sorted list of pointers, rather than a full sparse string. While this encoding has some similarities to first-quantized encodings [26] and sparse oracle encodings [2], we emphasize that we present a generic second-quantized encoding capable of performing any $O(1)$-local particle preserving rotation generated by creation/annihilation operators in the described complexity. Furthermore, this encoding will lay the groundwork for our later, more efficient encodings.

▶ **Theorem 7.** *There is a fermion data structure for strings of M qubits having Hamming weight at most F that uses $O(F \log M)$ qubits, with gate complexity $O(F \log M)$.*

We will constructively establish this theorem throughout Section 4.1 and 4.2 – in particular Lemma 9 and Lemma 10 demonstrate the requisite circuit complexities. An immediate corollary is that a similar statement holds for fermion encodings.

▶ **Corollary 8.** *There exists a fermion encoding of an M mode system having at most F fermions which uses $O(F \log M)$ qubits such that any k-local, particle preserving rotation can be implemented with $O(F \log M)$ gates.*

4.1 Algebraic Definition

We formally define the state encoding function $\mathcal{E}_s^{(1)}$ and give the algebraic properties it satisfies. Let $e_i \in \{0,1\}^M$ be a binary string that is all 0's, except for a single 1 at position i (using 1-based indexing). Consider strings of the form

$$x = e_{i_1} \oplus e_{i_2} \oplus ... \oplus e_{i_f}$$
$$i_1 < i_2 < ... < i_f$$

with $f \leq F$. Define $\mathcal{E}^{(1)} : \mathcal{D} \to \{0,1\}^{F\lceil \log(M+1) \rceil}$, where $\mathcal{D} \subset \{0,1\}^M$ is the set of strings of Hamming weight at most F, as

$$\mathcal{E}^{(1)}(x) = |i_1\rangle |i_2\rangle \cdots |i_f\rangle |\infty\rangle \cdots |\infty\rangle \tag{10}$$

where ∞ is a placeholder for "no fermion", and is the larger of any comparison with a non-∞. Concretely, we adopt the convention that ∞ is represented by a register of all 1's. We can interpret the output as a sorted array of f elements, each element (also referred to as register) of size $\lceil \log(M+1) \rceil$, and padded out to F entries by ∞'s. We will now describe the action of sign-rank and bit-flip queries on this list.

1. A $\mathsf{sgn\text{-}rank}(j, \mathbf{b})$ query should apply a -1 phase if there are an odd number of registers having values less than or equal to j. Otherwise, it should act like the identity.
2. A $\mathsf{bit\text{-}flip}(j, \mathbf{b})$ should insert a register $|j\rangle$ into the list if it is not present, and otherwise delete $|j\rangle$. This operation should maintain sorted order, and act as described so long as both input and output states have Hamming weight less or equal to F.

Observe that both of the aforementioned operations are reversible. This suffices to define a fermion data structure.

4.2 Efficient Circuits

We will utilize comparison circuits between unsigned integers as a black box. As described in Section 4.1, we will assume an upper limit F on the number of pointers we will ever need to store, and therefore only utilize this many registers.

▶ **Lemma 9.** *For the sorted list encoding described in Section 4.1, a* $\mathsf{sgn\text{-}rank}(p, \mathbf{b})$ *query has a circuit of* $O(F \log M)$ *gates and* $O(F \log M)$ *ancilla.*

Proof. Such a circuit should induce a -1 phase for every 1 present (in the original bit string) before position p. To achieve this, for each pointer register $|x\rangle$ we compute $x \leq p$, apply a Z to the outcome bit, then uncompute. Each comparison takes $O(\log M)$ gates to compare $O(\log M)$ size numbers, and there are $O(F)$ comparisons to make; the overall circuit complexity is therefore $O(F \log M)$. ◀

▶ **Lemma 10.** *For the sorted list encoding described in Section 4.1, a* $\mathsf{bit\text{-}flip}(p, \mathbf{b})$ *query has a circuit of* $O(F \log M)$ *gates and* $O(\log M)$ *ancilla.*

Proof. Such a circuit should add a pointer p to the list if there is none, otherwise it should remove the pointer p. At a high level, we will do this in three steps.

1. If p exists, move it to the last register
2. On the last register exchange p and ∞
3. If the last register is p, move it to the sorted position

To accomplish this reversibly, we first implement a reversible ordered-swap U_p. This operator swaps two registers if one is p and the other is $> p$. By chaining these together in ascending order, we will move p to the end if it exists. Exchanging classical states can be easily done, then chaining more U_p together in descending order will move p to sorted order if it was at the end. The full circuit requires $O(F)$ calls to subroutine U_p, and each takes $O(\log M)$ (from comparisons); exchanging two classical values on a register has negligible complexity. The overall complexity is therefore $O(F \log M)$. ◀

4.3 Lower Depth

A notable problem with the previously described encoding is the serial nature of the bit flip operator. In particular, the depth of the circuits as described is $O(F \log M)$, primarily due to sequentially "bubbling" a targeted register to the end of the list when implementing a $\mathsf{bit\text{-}flip}(p, \mathbf{b})$ operation. In this section, we describe a modification using $O(F \log M)$ ancillas that allows exponentially smaller depth, $O(\log F + \log \log M)$. Further, the buffer register idea developed here will prove useful when lowering depth in our succinct construction.

To achieve $O(\log F + \log \log M)$ depth, we will pad the encoded state with an $|\infty\rangle$ between registers and on either end. We will refer to these as buffer registers, in contrast with logical registers representing pointers. We will also pad the start with a $|-\infty\rangle$ and the end with $|\infty\rangle$ logical registers for convenience. Formally, consider strings of the form

$$x = e_{i^{(1)}} \oplus e_{i^{(2)}} \oplus ... \oplus e_{i^{(f)}}$$
$$i^{(1)} < i^{(2)} < ... < i^{(f)}$$

where $f < F$. Define $\mathcal{E}^{(2)} : \{0,1\}^M \to \{0,1\}^{O(F \log M)}$ as

$$\mathcal{E}^{(2)}(x) = |-\infty\rangle\,|\infty\rangle\,|i_l^{(1)}\rangle\,|\infty\rangle\,|i_l^{(2)}\rangle\,|\infty\rangle \cdots |\infty\rangle\,|i_l^{(F)}\rangle\,|\infty\rangle \cdots |\infty\rangle\,, \tag{11}$$

where every other register is a buffer (coloured blue, but this is purely conceptual; the same data is stored in each). This leaves $O(F \log M)$ qubits, as the buffer registers are just a constant factor overhead. The correct action of sgn-rank and bit-flip exactly mirrors that of Section 4 on the logical registers (excluding the initial padded $-\infty$), except now we must also leave the buffer registers invariant after the computation is finished.

▶ **Lemma 11.** *Under the list-with-buffers encoding described in Section 4.3, a* sgn-rank$(p, \mathbf{b})$ *query has a circuit of* $O(\log \log M)$ *depth,* $O(F \log M)$ *gates, and* $O(F \log M)$ *ancilla.*

Proof. This operation can be done in a similar way to Section 4.2. We again apply a conditional phase to each logical register (note we do not do this on any buffer register, nor the padded $|-\infty\rangle$ at the beginning). The gate count $O(F \log M)$ follows from the analysis in Lemma 9, and one can see that the depth is fully determined by the depth of a comparison. By simple facts about comparator circuits, these can be done in $O(\log \log M)$ depth using linearly many ancillae, so the overall depth is $O(\log \log M)$. ◀

▶ **Lemma 12.** *Under the list-with-buffers encoding described in Section 4.3, a* bit-flip$(p, \mathbf{b})$ *query has a circuit of* $O(\log F + \log \log M)$ *depth,* $O(F \log M)$ *gates, and* $O(F \log M)$ *ancilla.*

Proof. We are promised that whenever an insertion is made, the last element will be an ∞. Conversely, whenever a deletion is made, the element added to the end will be an ∞. This prevents us from needing to perform a full cycle, instead relying on the buffer registers to provide ∞'s wherever needed. At a high level, it suffices to do the following:

1. Compute a flag f_{del} depending on whether p appears in the logical registers, and fan it out to all registers.
2. If p is not present ($f_{del} = \top$), insert p at the buffer register between it's immediate predecessor and successor.
3. If p is present, shuffle registers $> p$ upwards one position, and move p to it's preceding buffer. Otherwise, shuffle all registers $> p$ downwards one position, and move p out of it's preceding buffer.
4. If p was present ($f_{del} = \bot$), remove p from the buffer between p's predecessor and successor.
5. Uncompute f_{del}.

Note that steps (2) and (4) do not need to explicitly be controlled by f_{del} due to the assumed structure of the data. In step (3) however, the direction in which to shuffle depends on whether p is present or not, so a light cone argument implies that the local operations must depend on whether p is present. It is worth pointing out that, except for the fan-in/fan-out of f_{del}, all operations are nearest-register in a one dimensional layout, which could aid implementation on a quantum computer with geometric constraints.

Using linearly many ancilla registers, a comparison between n bit integers can be done in depth $\log n$. Swaps between registers can be done in $O(1)$ depth, the largest comparison is between $O(\log M)$ bit integers, and a single bit can be fanned out to F positions (one for each register) in depth $O(\log F)$. The total depth is therefore $O(\log F + \log \log M)$ (or $O(\log \log M)$ with unbounded fan-in and fan-out), using $O(F \log M)$ ancillae. Each register of size $\log M$ is acted on by only a constant number of comparison/swaps, so the gate complexity is $O(F \log M)$. ◀

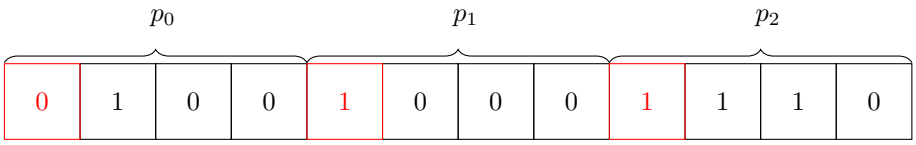

Figure 1 An illustration of the redundancy in a sorted list, for a list of 3 things out of a universe of 16 things. The red bits are non-decreasing, with possibilities: $000, 001, 011, 111$. Four possibilities means 2 bits suffice, rather than 3.

5 Achieving Succinctness

Recall that the information theoretic limit for encoding F fermions in M modes is $\mathcal{I} := \lceil \log \binom{M}{F} \rceil$ qubits. From Stirling's approximation we have

$$\mathcal{I} = F \log \frac{M}{F} + O(F),$$

so the encoding in Section 4.1 is redundant – for instance when $F = \sqrt{M}$, approximately half of the qubits are unnecessary. Here we present an encoding in which only a sublinear $O(F)$ many qubits are unnecessary (there are known constructions that achieve the *exact* information theoretic minimum $\mathcal{I}$ [14, 23], but the gate complexity becomes $M^{O(F)}$). For the example of $F = \sqrt{M}$, this new encoding has only a negligible $o(1)$ fraction of unnecessary qubits, i.e. it is roughly twice as space efficient as in Section 4. These techniques make crucial use of the fact that the list is not anti-symmetrized.

5.1 Succinct encoding

We formally state the main theorem of this section below.

▶ **Theorem 13.** *There is a fermion data structure for strings of M qubits having Hamming weight at most F that uses $\mathcal{I} + O(F)$ qubits, with gate complexity $O(F \log M)$.*

Such an encoding is constructed in the remainder of this section, with Lemma 15 and Lemma 16 demonstrating that it satisfies the requisite efficiencies. Similar to Corollary 8, this implies a fermion to qubit mapping with the same complexities.

▶ **Corollary 14.** *There exists a fermion encoding of an M mode system having at most F fermions which uses $\mathcal{I} + O(F)$ qubits such that any k-local, particle preserving rotation can be implemented with $O(F \log M)$ gates.*

To reduce the number of necessary qubits, the key insight is noting that the most significant $G := \lceil \log F \rceil$ qubits in each register of the encoding in Section 4.1 are not pulling their weight. They use $F \cdot G = \Omega(F \log F)$ qubits to encode F integers in the range 1 to F, but they are in *sorted order*, reducing the number of possible sequences – see Figure 1 for an illustration. We can instead store the same data (the most significant bits) in $O(F)$ space as a bit string where $2^G - 1 = O(F)$ zeroes delimit 2^G bins, and F ones correspond to F fermions. The i-th bin is the space from the i-th zero (or the start of the bit string) to the $i+1$-st zero (or the end of the bit string), and the number of ones in that range indicate elements with most significant bits i. This idea is commonly referred to as the stars and bars method.

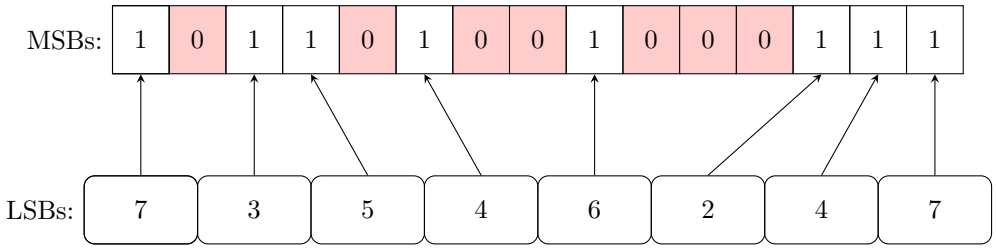

Figure 2 Illustrative example of the succinct representation, with parameters $F = 8$, $M = 63$ and fermions at $[7, 11, 13, 20, 38, 58, 60, \infty]$; recall the convention that the largest storable value (63) is identified with ∞. Red registers demarcate "bins" of different MSB values, out of 8 possible values. The LSBs then only require 3 qubits each, so the total storage is $15 + 24 = 39$ qubits. Using first quantization/sorted list encodings would use $8 \cdot \lceil \log 63 \rceil = 48$ qubits, and Jordan-Wigner or similar second quantized encodings would require 63 qubits.

The remaining least-significant bits can be stored in the usual way, where the order depends on the value determined by both least significant and most significant parts. So long as this order is maintained, the least-significant bits can be paired with the corresponding most-significant bits by the order in which they appear. This saves $F\lceil \log F \rceil$ qubits and adds $O(F)$ many, so the total qubit usage is

$$\begin{aligned}\text{Space}(M, F) =& F\lceil \log(M+1) \rceil - F\lceil \log F \rceil + O(F) \\ =& F \log \frac{M}{F} + O(F) \\ =& \mathcal{I} + O(F).\end{aligned}$$

To define the encoding formally, for strings of the form

$$\begin{aligned} x &= e_{i^{(1)}} \oplus e_{i^{(2)}} \oplus \ldots \oplus e_{i^{(F)}} \\ i^{(1)} &< i^{(2)} < \ldots < i^{(f)} \end{aligned}$$

for $f < F$, let i_m denote the G most significant bits of i, and i_l denote the remaining least significant bits. Let r_j be the number of pointers having most significant bits equal to j, i.e. the number of values of k such that $i_m^{(k)} = j$. Now define $\mathcal{E}^{(3)} : \{0,1\}^M \to \{0,1\}^{\mathcal{I}+O(F)}$ as

$$\mathcal{E}^{(3)}(x) = |i_l^{(1)}\rangle\, |i_l^{(2)}\rangle \cdots |i_l^{(f)}\rangle\, |\infty_l\rangle \cdots |\infty_l\rangle\, ||1^{r_0}01^{r_1}0 \ldots 01^{r_{2^G-1}} \tag{12}$$

Where $||$ is a conceptual divider between information about the most and least significant bits, and 1^n denotes n many 1's. The bits to the left of $||$ store the least significant bits of each pointer, and are referred to as l or LSBs, and i-th register in this section is l_i. The bits to the right of $||$ are referred to as m or MSBs and store the most significant bits of each pointer. Define $\mathcal{E}_s^{(3)} : \mathcal{H}_{2^M} \to \mathcal{H}_{2^{\mathcal{I}+O(F)}}$ as the unique linear extension of $\mathcal{E}^{(3)}$. A simple example of this encoding is shown in Figure 2.

For correctness, it suffices if encoded bit-flip and sgn-rank operators satisfy the rules described in Section 4.1, on the list implicitly represented by our succinct string. In particular, the encoded $\mathsf{bit\text{-}flip}(p, \mathbf{b})$ operator should unitarily insert/delete p from the sorted list defined by the information stored in the MSBs and LSBs, and update this data accordingly. The $\mathsf{sgn\text{-}rank}(p, \mathbf{b})$ operator should apply a minus phase if and only if there are an odd number of elements less or equal to p in the sorted list.

5.2 Efficient circuits

We now turn to constructing efficient circuits for the necessary operations on this succinct encoding. We note that retaining our space complexity advantage limits the number of ancillas that can be used to perform our operations to be $O(F)$ (and, naturally, we require these ancillas to be uncomputed by the end of each operation).

▶ **Lemma 15.** *Under the succinct list encoding $\mathcal{E}_s^{(3)}$ described in Section 5.1, a* $\mathsf{sgn\text{-}rank}(p, \mathbf{b})$ *query has a circuit of $O(F \log M)$ gates, and $O(F)$ ancilla.*

Proof. At a high level, this operation can be performed as follows:

1. Scan the most significant bits m to determine a range $[p_{start}, p_{end})$ of least significant bit registers whose most significant bits match p
2. Use this range to implement comparisons $\leq p$ on all registers
3. Phase the result of this comparison with a Z gate
4. Uncompute comparisons, then p_{start} and p_{end}

Scanning the register m takes $O(F)$ iterations costing $O(\log F)$ each. Each comparison afterwards costs $O(\log M/F)$ gates, and there are $O(F)$ many to make, so the total gate complexity is $O(F \log M)$. The only space used beyond comparisons are registers of size $O(\log F)$, satisfying the ancilla requirement. ◀

▶ **Lemma 16.** *Under the succinct list encoding $\mathcal{E}_s^{(3)}$ described in Section 5.1, a* $\mathsf{bit\text{-}flip}(p, \mathbf{b})$ *query has a circuit of $O(F \log M)$ gates, and $O(F)$ ancilla.*

Proof. Comparisons require information from the most significant bits, which is gathered first. Following this, analogous to the approach in Lemma 10 we will bubble the target element to the end if present, exchange with ∞, and then bubble back. At a high level:

1. Compute and store a pointer t to the target position in the list of F elements, defined as the first register with value $\geq p$ (based on both it's most and least significant bits).
2. Compute a flag f_{del}, which is on if and only if p is present in the list.
3. If f_{del}, shuffle l_t to the end of l.
4. Exchange $\infty_l \leftrightarrow p_l$ on the last register of l.
5. If not f_{del}, shuffle the end of l to t.
6. If f_{del}, shuffle the $(t+p)$-th bit of m to the end
7. If not f_{del}, shuffle the last bit of m to the $(t+p)$-th position.
8. Uncompute f_{del} and t.

Steps (1) and (2) can be done in $O(F \log M)$ gates by computing temporary p_{start}, p_{end} ranges of indices in l which have the same most significant bits as p, implementing comparisons using these, then uncomputing. Steps (3), (4), (6), (7) are simple swap ladders which can be performed in $O(F \log M)$ gates, and Step (5) has negligible complexity. Step (8) has the same cost as steps (1), (2), so the overall gate complexity is $O(F \log M)$. Again, the only additional space stored is $O(\log F)$ size pointers p_{start} and p_{end}, satisfying the ancilla requirement. ◀

6 Applications

In this section we give a more detailed comparison of our encoding to prior encodings, and outline scenarios where ours outperforms prior work. We will focus on the succinct construction, e.g. Theorem 1, as it is likely the more practical of the two constructions.

For concreteness we consider the parameter regime where the number of fermions is polynomially smaller than the number of modes, i.e. $F = M^c$ for some constant $0 < c < 1$, though we note that our encoding is efficient outside this regime as well. This regime is

well motivated for applications such as quantum chemistry, where the number of modes is preferred to be as large as possible to accurately model the continuum, but algorithm's polynomial scaling with M prevents M from being exponentially larger than F.

The primary application we find is in randomized simulation algorithms such as qDRIFT [9], where our encoding gives a polynomial space savings with only logarithmic depth overhead. We also show that our encoding allows for an optimal implementation of a second quantized SELECT subroutine, used in linear combination of unitary methods [10], no matter how M and F relate.

6.1 Space Efficient Randomized Simulation

Randomized product formulas like the qDRIFT algorithm [9] are strong candidates where our encoding will be useful, as the complexity of such algorithms depend heavily on the fermion encoding used. The qDRIFT algorithm requires $O((\lambda t)^2/\epsilon)$ fermionic rotation operators to evolve for time t within error ϵ, where λ is the sum of the magnitude of Hamiltonian coefficients when written in a second quantized basis. The scaling with λ is preferable to scaling with M when the magnitude of coefficients varies drastically, as is often the case for quantum chemistry Hamiltonians.

▶ **Theorem 17.** *For a particle preserving local fermionic Hamiltonian H on an M mode system, there is a second quantized algorithm for evolving an F fermion state to time t using space $\mathcal{I}+O(F)$, with gate complexity $O\left(\epsilon^{-1}(\lambda t)^2\mathcal{I}\right)$, and circuit depth $O\left(\epsilon^{-1}(\lambda t)^2 \log M \log\log M\right)$, where λ denotes the sum of magnitudes of coefficients in H and ϵ denotes the target diamond-norm error.*

This theorem follows straightforwardly from using the qDRIFT algorithm [9] with our Theorem 1 In the regime discussed above, the space complexity of this algorithm is

$$\mathcal{I} + O(F) = O(M^c \log M^{1-c}) = O(M^c \log M),$$

with gate complexity

$$O\left(\epsilon^{-1}(\lambda t)^2\mathcal{I}\right) = O(\epsilon^{-1}(\lambda t)^2 M^c \log M),$$

and circuit depth

$$O\left(\epsilon^{-1}(\lambda t)^2 \log M \log\log M\right).$$

We now compare these complexities against those achieved by other second-quantized encodings in our parameter regime.

6.1.1 Space Complexity

The most space-efficient prior second quantized encodings in this regime that are amenable to randomized simulation methods are the qubit efficient [23] and permutation basis [14], but these encodings would incur an exponential (M^F) gate and depth overhead. Barring these encodings, the next best are the segment code [25, 24], and the optimal degree code [19]. We use polynomially less space than the former ($O(M^c \log M)$ vs. $\Omega(M)$) and quadratically less than the latter ($O(F \log M)$ vs. $\Omega(F^2 \log^4 M)$).

6.1.2 Circuit Complexity

When $c < 1/2$, the prior best second quantized encoding in terms of space usage (barring codes with exponential gate overhead) is the optimal degree code [19]. Compared to this encoding, along with better space efficiencies we also improve on the $O(F^2 \log^5 M)$ gate complexity of fermion operations. Our scaling of $O(\mathcal{I})$ (with $\mathcal{I} < F \log M$) is always quadratically better in dependence on F, and at least four powers better in dependence on $\log M$. Further, while the depth is not analyzed in the optimal degree encoding a simple counting argument implies that it is at least $\Omega((\lambda t)^2 \log M/\epsilon)$, which is at most a negligible $O(\log \log M)$ better than our encoding. Compared to other space efficient second-quantized encodings, we give at least a quadratic improvement in circuit complexity while also removing many log factors.

Allowing space inefficient encodings, the most gate efficient prior encoding in our regime is the Bravyi-Kitaev encoding [7], which uses $O(\log M)$ gates to implement a fermionic rotation. This is the encoding used in the original resource estimates for the qDRIFT algorithm, making it a natural competitor. When the order of fermionic rotations is selected randomly, the Bravyi-Kitaev encoding of these operators cannot be significantly parallelized. The depth for full simulation will generically be $\Omega((\lambda t)^2/\epsilon)$ (and potentially larger if fermion operations take super-constant depth to implement), which is only a logarithmic factor lower than in our encoding. Furthermore, the Bravyi-Kitaev encoding requires polynomially more qubits ($O(M)$ versus $O(M^c \log M)$) than our encoding in this regime – though fewer gates ($O(\log M)$ versus $O(M^c \log M)$). This trade-off is therefore more beneficial the smaller c is.

6.2 Optimal SELECT Subroutine

In the context of post-Trotter methods, our encoding method can be relevant not only for saving space but also for saving gates. We illustrate this by giving a provably optimal implementation of the SELECT subroutine, a key component in the linear combination of unitaries algorithm for Hamiltonian simulation [10]. This algorithm involves expressing the Hamiltonian as a linear combination of unitaries

$$H = \alpha_1 U_1 + ... + \alpha_q U_q,$$

then block encoding the Hamiltonian using a PREPARE oracle that creates a superposition over every number in $j \in [q]$, weighted by amplitude α_j, in conjunction with as a so-called SELECT oracle. The SELECT oracle has a control wire indicating a number $j \in [q]$, and conditioned on j performs the unitary U_j on the physical state – this oracle is the only part whose implementation is dependent on the fermion encoding used. One way to split a k-local second quantized fermionic Hamiltonian into a linear combination of unitaries is to split every a_j into Majoranas $a_j = \gamma_{2j-1} + i\gamma_{2j}$. The γ_j operators are unitary, meaning the full Hamiltonian is now a linear combination of unitaries. Using our encoding, we can perform a SELECT oracle in the optimal complexity. Specifically, we show how to perform an k-local product of Majorana operators, given k index wires denoting which Majoranas appear in the product. This translates to a sequence of $O(k)$ many sgn-rank and bit-flip operations, so in particular it suffices to do a single $\textsf{sgn-rank}(j, \mathbf{b})$ or $\textsf{bit-flip}(j, \mathbf{b})$ operator controlled on an index wire $|j\rangle$.

▶ **Theorem 18.** *Using the encoding described in Theorem 1, we can perform a coherently controlled* $\textsf{sgn-rank}(|j\rangle, \mathbf{b})$ *or* $\textsf{bit-flip}(|j\rangle, \mathbf{b})$ *with* $O(\mathcal{I})$ *gates.*

Proof. Consider the M fermion Hamiltonian

$$H^{(m)} = \alpha_1 U_1^{(m)} + ... + \alpha_q U_q^{(m)}$$

where every $U_i^{(m)}$ is a product of Majorana operators. When the fermionic Hamiltonian H is k-local, then every $U_j^{(m)}$ appearing in $H^{(m)}$ has k-many Majorana operators. It follows that we can implement the action of $U_j^{(m)}$ within the encoding described in Theorem 1 with complexity $O(\mathcal{I})$. We can further coherently condition which $U_j^{(m)}$ is performed using a control register (see the full version for details), introducing at most a constant factor overhead. For this SELECT oracle we therefore obtain circuits of size $O(\mathcal{I})$. Note that in general there are many ways to split a Hamiltonian into a linear combination of unitaries, and we leave to future work whether SELECT has an efficient implementation under alternative choices. ◀

As argued above, this is sufficient to implement a SELECT oracle in the same complexity. We now turn to lower bounds.

▶ **Theorem 19.** *Using any fermion to qubit mapping, a coherently controlled, second-quantized, k-local particle preserving fermion operation (a SELECT oracle) requires $\Omega(\mathcal{I})$ gates.*

Proof. The lower bound for representing F fermions in M modes is $\mathcal{I}$ qubits, due to a straightforward counting argument. Suppose there were a generic implementation of the second quantized SELECT oracle of the aforementioned kind which used $\lfloor \frac{1}{2}(\mathcal{I}-1) \rfloor$ one- and two-qubit gates, and implemented any k-local fermion term for a $k > 1$. It follows that such an oracle would act on at most $\mathcal{I}-1$ qubits, so consider keeping only the $\mathcal{I}-1$ qubits it acts on. We are then left with a fermion encoding on $\mathcal{I}-1$ qubits, as by assumption the SELECT oracle is capable of performing arbitrary pairs of Majorana operators; a contradiction. Hence, any generic implementation of the SELECT oracle requires $\Omega(\mathcal{I})$ gates, matching our upper bound up to constant factors. ◀

To compare this result against existing second quantized encodings, note that in all prior encodings (summarized in Table 1) the complexity of a coherently controlled fermion operator is at least linear in the number of qubits, or exponential in the case of prior succinct encodings. For every gate-efficient encoding, the number of qubits is polynomially larger than $\mathcal{I}$ in our regime $F = M^c$, from which it follows that the gate complexity is at least polynomially larger than $\mathcal{I}$. Concretely, when $c < 1/2$ the best prior implementation of the select oracle is the optimal degree encoding, with gate complexity $\Omega(F^2 \log^5 M)$. Our implementation of complexity $O(\mathcal{I})$ where $\mathcal{I} < F \log M$ is quadratically better in F dependence, and at least four powers better in $\log M$ dependence. We note also that in the parameter regime $c > 1/2$ our encoding maintains at least a factor $\tilde{O}(M^{1-c})$ advantage in gate complexity over prior work, giving a polynomial advantage over the whole regime.

6.3 Comparison to First Quantized Encodings

First quantized algorithms maintain a list of pointers representing fermion positions similar to the encoding described in Section 4, but antisymmetrize the state over all possible permutations. The fermion phases then arise from the exchange statistics of the registers, allowing for a different class of efficient operations. Comparisons against first quantization are subtle, as algorithms within these two paradigms can have drastically different complexities for simulating similar systems. In general one would expect first quantized algorithms to

outperform second quantized algorithms when F is extremely small compared to M, but there is evidence that second quantized algorithms can remain advantageous when F is polynomially smaller than M [27]. This motivates the parameter regime we consider here.

6.3.1 Space Complexity

Using Stirling's approximation we can rewrite

$$\mathcal{I} = F \log \frac{M}{F} + O(F). \tag{13}$$

In our parameter regime of interest, we therefore have

$$\mathcal{I} = (1-c) \cdot F \log M + O(F) \tag{14}$$

Note that the $O(F)$ term is asymptotically less than the $(1-c) \cdot F \log M$ term, and as a corollary the asymptotic space usage of our encoding $(\mathcal{I} + O(F))$ is a factor $1 + o(1)$ from optimal, i.e. it is succinct. Standard first-quantized encodings [26] use $F\lceil \log M \rceil$, which is asymptotically a factor $\frac{1}{1-c}$ above $\mathcal{I}$, which is not succinct. Our encodings saves more space the closer c is to 1.

6.3.2 Gate Complexity

Both our representation and first-quantized representations support efficient (i.e. at most linear in the number of qubits) fermion operations of the first-quantized or second-quantized variety, respectively. Differences in gate complexity, therefore, stem primarily from differences in algorithms built in a first-quantized versus second-quantized picture. Many of the algorithms in these different paradigms are designed for different contexts, making a fair comparison difficult.

7 Conclusion and Open Problems

We presented two second-quantized fermion encodings which are almost exactly optimal in terms of space usage, yet allows gate efficient and low depth second-quantized fermion rotations. In the context of randomized simulation algorithms, the succinct encoding allows for a polynomial space savings with only a logarithmic depth overhead. Along the way, we presented a more simple succinct encoding which did not achieve low depth, as well as a low depth encoding that did not achieve succinctness, though both of these encodings may be of independent interest due to their simplicity. This work leaves open the following questions.

1. In first quantization, a list of positions is stored in an antisymmetrized wavefunction with $F\lceil \log M \rceil$ qubits in standard encodings, allowing for efficient first quantized fermion operations. Can such a representation be made succinct while still antisymmetrizing?
2. Is there an efficient fermionic Fourier transform circuit for our encodings?
3. Can the qubit count be improved beyond $\mathcal{I} + O(F)$ without exceeding gate complexity $O(\mathcal{I})$? We conjecture a lower bound on the qubits required to maintain this gate complexity is $\mathcal{I} + \Omega(F)$.
4. When compiled to a universal gate set like $\mathrm{CNOT}, \mathrm{H}, T$, the complexities of each gate is the same as the overall complexity in our encodings. In many error correcting codes certain gates are harder than others (for instance T gates are more expensive in a surface code), motivating analysis of each gate individually. Can the number of T gates be reduced in this construction?
5. Can the degree of the $\mathsf{poly}(M)$ circuit complexity scaling in our implicit encoding be improved?

References

1 Ryan Babbush, Dominic W. Berry, Jarrod R. McClean, and Hartmut Neven. Quantum simulation of chemistry with sublinear scaling in basis size. *npj Quantum Information*, 5(1):92, November 2019. doi:10.1038/s41534-019-0199-y.

2 Ryan Babbush, Dominic W Berry, Yuval R Sanders, Ian D Kivlichan, Artur Scherer, Annie Y Wei, Peter J Love, and Alán Aspuru-Guzik. Exponentially more precise quantum simulation of fermions in the configuration interaction representation. *Quantum Science and Technology*, 3(1):015006, December 2017. doi:10.1088/2058-9565/aa9463.

3 D. W. Berry, A. M. Childs, and R. Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 792–809, Los Alamitos, CA, USA, October 2015. IEEE Computer Society. doi:10.1109/FOCS.2015.54.

4 Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Simulating hamiltonian dynamics with a truncated taylor series. *Phys. Rev. Lett.*, 114:090502, March 2015. doi:10.1103/PhysRevLett.114.090502.

5 Dominic W. Berry, Andrew M. Childs, Yuan Su, Xin Wang, and Nathan Wiebe. Time-dependent Hamiltonian simulation with L^1-norm scaling. *Quantum*, 4:254, April 2020. doi:10.22331/q-2020-04-20-254.

6 Sergey Bravyi, Jay M. Gambetta, Antonio Mezzacapo, and Kristan Temme. Tapering off qubits to simulate fermionic Hamiltonians, 2017. arXiv:1701.08213.

7 Sergey B. Bravyi and Alexei Yu Kitaev. Fermionic Quantum Computation. *Annals of Physics*, 298(1):210–226, May 2002. doi:10.1006/aphy.2002.6254.

8 Harry Buhrman, Bruno Loff, Subhasree Patro, and Florian Speelman. Memory compression with quantum random-access gates. In *Theory of Quantum Computation, Communication, and Cryptography*, 2022. URL: https://api.semanticscholar.org/CorpusID:247411405.

9 Earl Campbell. Random compiler for fast Hamiltonian simulation. *Phys. Rev. Lett.*, 123:070503, August 2019. doi:10.1103/PhysRevLett.123.070503.

10 Andrew M. Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Info. Comput.*, 12(11–12):901–924, November 2012. doi:10.26421/QIC12.11-12-1.

11 Charles Derby, Joel Klassen, Johannes Bausch, and Toby Cubitt. Compact fermion to qubit mappings. *Phys. Rev. B*, 104:035118, July 2021. doi:10.1103/PhysRevB.104.035118.

12 Anna Gál and Peter Bro Miltersen. The cell probe complexity of succinct data structures. *Theoretical Computer Science*, 379(3):405–417, 2007. Automata, Languages and Programming. doi:10.1016/j.tcs.2007.02.047.

13 Craig Gidney. Quantum dictionaries without QRAM, 2022. arXiv:2204.13835.

14 Brent Harrison, Dylan Nelson, Daniel Adamiak, and James Whitfield. Reducing the qubit requirement of Jordan-Wigner encodings of N-mode, K-fermion systems from N to $\lceil \log_2 \binom{N}{K} \rceil$, November 2022. arXiv:2211.04501.

15 Guy Joseph Jacobson. *Succinct static data structures*. PhD thesis, Carnegie Mellon University, USA, 1988. AAI8918056.

16 Samuel Jaques and Arthur G. Rattew. Qram: A survey and critique, 2023. arXiv:2305.10310.

17 P. Jordan and E. Wigner. On the Paulian prohibition of equivalence. *Z. Physik*, 47:631–651, 1928. doi:10.1007/BF01331938.

18 Ivan Kassal, Stephen P. Jordan, Peter J. Love, Masoud Mohseni, and Alán Aspuru-Guzik. Polynomial-time quantum algorithm for the simulation of chemical dynamics. *Proceedings of the National Academy of Sciences*, 105(48):18681–18686, 2008. doi:10.1073/pnas.0808245105.

19 William Kirby, Bryce Fuller, Charles Hadfield, and Antonio Mezzacapo. Second-quantized fermionic operators with polylogarithmic qubit and gate complexity. *PRX Quantum*, 3:020351, June 2022. doi:10.1103/PRXQuantum.3.020351.

20 Guang Hao Low and Isaac L. Chuang. Optimal hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.*, 118:010501, January 2017. doi:10.1103/PhysRevLett.118.010501.

21 Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Qubitization. *Quantum*, 3:163, July 2019. doi:10.22331/q-2019-07-12-163.

22 Gonzalo Navarro. *Compact Data Structures: A Practical Approach*. Cambridge University Press, 2016.

23 Yu Shee, Pei-Kai Tsai, Cheng-Lin Hong, Hao-Chung Cheng, and Hsi-Sheng Goan. Qubit-efficient encoding scheme for quantum simulations of electronic structure. *Phys. Rev. Res.*, 4:023154, May 2022. doi:10.1103/PhysRevResearch.4.023154.

24 Mark Steudtner. *Methods to simulate fermions on quantum computers with hardware limitations*. PhD thesis, Leiden University, 2019.

25 Mark Steudtner and Stephanie Wehner. Fermion-to-qubit mappings with varying resource requirements for quantum simulation. *New Journal of Physics*, 20(6):063010, June 2018. doi:10.1088/1367-2630/aac54f.

26 Yuan Su, Dominic W. Berry, Nathan Wiebe, Nicholas Rubin, and Ryan Babbush. Fault-tolerant quantum simulations of chemistry in first quantization. *PRX Quantum*, 2:040332, November 2021. doi:10.1103/PRXQuantum.2.040332.

27 Yuan Su, Hsin-Yuan Huang, and Earl T. Campbell. Nearly tight Trotterization of interacting electrons. *Quantum*, 5:495, July 2021. doi:10.22331/q-2021-07-05-495.

28 Matthias Troyer and Uwe-Jens Wiese. Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations. *Phys. Rev. Lett.*, 94:170201, May 2005. doi:10.1103/PhysRevLett.94.170201.

29 F Verstraete and J I Cirac. Mapping local Hamiltonians of fermions to local Hamiltonians of spins. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09012, September 2005. doi:10.1088/1742-5468/2005/09/P09012.

30 James D. Whitfield, Vojtěch Havlíček, and Matthias Troyer. Local spin operators for fermion simulations. *Phys. Rev. A*, 94:030301, September 2016. doi:10.1103/PhysRevA.94.030301.