Technical note: Spin-fermion transformations using parity trees

J. D. Whitfield
Department of Physics and Astronomy, Dartmouth College, Hanover, New Hampshire 03755, USA

Quantum simulation of fermionic systems using qubits will play an important role in current and emerging quantum computing architectures. Limitations of small scale quantum computers will help inform the roadmap to the future and here we present new tools for creating qubit operators for fermionic systems. In this technical note, we give an introduction and a summary of our generalized approach to Jordan-Wigner and Bravyi-Kitaev transforms and beyond. In particular, the parity tree forms the crux of our general approach. The most important parity tree is the Fenwick Tree as detailed in our last publication. The corresponding code is available on GitHub at https://github.com/jdwhitfield/Julia-qFenwick.

In this short tutorial, we concern ourselves with the spin and fermion algebras motivated by the desire to simulate fermionic systems using controllable spin systems i.e. quantum computers. If each spin is two dimensional i.e. qubits, then operators on the system are describable using the tensor product of local two dimensional matrices. Two common bases for describing the degrees of freedom of the local spin system are the Hermitian and unitary Pauli spin matrices (I, X, Y, Z) and the basis of the matrix elements themselves $(S_\pm = (X \pm iY)/2, P_{00} = (I + Z)/2, P_{11} = (I - Z)/2)$. The spin algebra imposes that $XY = iZ$, $ZX = iY$, and $YZ = iX$.

All fermionic systems must be antisymmetric. To use spin systems to perform the simulation, this antisymmetry must be imposed. There are a number of ideas in the literature for performing this task as review in our most recent paper (see IV, Further Reading). The Jordan-Wigner transform, the Bravyi-Kitaev transform, and the first quantization encoding are well-discussed examples.

Antisymmetric states have the property that $P_\sigma \Psi = \mathrm{sgn}(\sigma)\Psi$ for $\sigma \in S_N$. If P_σ can be decomposed into a even (odd) number of transpositions, then $\mathrm{sgn}(\sigma)$ is $+1$ (-1). First quantization is predicated on the fact that $P_\sigma \mathcal{A} = \mathrm{sgn}(\sigma)\mathcal{A}$ for the antisymmetrizer $\mathcal{A} = \sum \mathrm{sgn}(\sigma) P_\sigma$.

The first quantization encodings explicitly antisymmetrizes N registers each labelling one of the M occupied single-particle wave functions (atomic orbitals, plane waves, etc.) :

$$\mathcal{A}|\Phi\rangle = \sum_{\sigma}^{S_N} P_\sigma |i_1 i_2 ... i_N\rangle \tag{1}$$

$$= \sum_{\sigma}^{S_N} |i_{\sigma(1)} i_{\sigma(2)} ... i_{\sigma(N)}\rangle \tag{2}$$

Here i_k takes values from $1...M$ leading to a state space of dimension M^N before antisymmetrization. Because the antisymmetrization is explicitly done by swapping the N registers, the order of the occupied labels is not important; All orderings will be projected into the completely antisymmetric space of dimension $\binom{M}{N}$. The second quantization methods are based around an alternative approach to storing the occupancies.

Instead of treating all orderings of orbital labels on the same footing, we could pick a canonical ordering (as in the ordering of a deck of cards) and store M bits. That is to say, instead of states of the form $\{|i_1 i_2 ... i_N\rangle : i_j = 1, 2, ..., M\}$, second quantization considers states of the form $\{|K\rangle = |k_1 k_2 ... k_M\rangle : k_j = 0, 1\}$.

We can also understand antisymmetry in second quantization from an algebraic perspective. Let us define an fermionic algebra of operators $\{a_j\}$ using the canonical anti-commutation relations $[a_j^\dagger, a_k]_+ = \delta_{jk}$ and $[a_j, a_k]_+ = 0$. Also define $|\Omega\rangle$ as a fixed state such that $a_j|\Omega\rangle = 0$ for all j. We can show this algebra generates antisymmetric states with $|K\rangle = \prod_{i=1}^M (a_i^{k_i})^\dagger |\Omega\rangle$. If we have a state $a_{i_1}^\dagger a_{i_2}^\dagger ... a_{i_N}^\dagger |\Omega\rangle$ with an arbitrary sequence of indices i_j, we can reorder back to the canonical indexing using the algebra $a_m^\dagger a_n^\dagger = -a_n^\dagger a_m^\dagger$. This reordering requires a phase factor $\mathrm{sgn}(\sigma)$ corresponding to the permutation P_σ needed to reach the canonical ordering. Similarly,

$$\begin{aligned} a_j^\dagger |K\rangle &= a_j^\dagger \prod_{i=1}^M (a_i^{k_i})^\dagger |\Omega\rangle \\ &= \left(\prod_{i=1}^{j-1} (-a_i^{k_i})^\dagger\right) \left[a_j^\dagger (a_j^{k_j})^\dagger\right] \left(\prod_{i=j+1}^{M} (a_i^{k_i})^\dagger\right) |\Omega\rangle \\ &= \left[\delta_{k_j,0}\right] (-1)^{\Gamma_j} \prod_{i=1}^M (a_i^{k_i})^\dagger |\Omega\rangle \\ &= \delta_{k_j,0} (-1)^{\Gamma_j} |K'\rangle \end{aligned} \tag{3}$$

In eq. (3), $\Gamma_j = \sum_{i=1}^{j-1} k_i$ and K' is such that $k'_i = k_i$ for $i \neq j$ and $k'_j = 1$.

Note that ordering the orbital labels is fixed by the linear indexing of the bits k_i. Thus, the states cannot be projected into antisymmetric states by permuting registers as in the first quantization idea. Second quantization avoids explicitly constructing $\mathcal{A}$ by explicitly computing the phase factor $\mathrm{sgn}(\sigma)$ as necessary.

Here, we consider a broad family of algorithmic techniques to compute the phase factor in second quantization. The characterization discussed here is based on the notion of a parity tree introduced in the next section. The parity trees encapsulate many of the techniques in the literature as explained in Section II. Lastly, we give concrete examples of the mapping on small systems. Source

code in the Julia language is given on GitHub.

I. PARITY TREES

Parity tree structures are useful when both occupancies (k_i) and prefix sums ($\Gamma_j = \sum_{i=1}^{j-1} k_i$) are required. In this section we will define the general parity tree and the specific class we consider here. Further, we will define a number of sets that play an instrumental role in extracting prefix sums and antisymmetrical phases later.

The parity tree has several fermionic occupancies ($\{k_i \in \{0,1\}\}_{i=1}^{M}$) summed to give a binary value stored at a particular node, $t_j = \sum_i \beta_{ij} k_i$, of the tree. This β matrix representation was introduced by Seeley et al. Here, we focus trees where each node has at most one parent but may have an arbitrary number of children. We don't believe this requirement is fundamental but it does capture the previous known algorithmic encodings.

As we pointed out earlier, we assume each node has at most one parent. The parent of the kth tree node can be stored in bin k of a linear tree list e.g. $\mathcal{T}_1 = [2,3,1]$ would indicate that node one is a child of node two, two is a child of three and three is a child of node one. If the node has no parent, then a marker is stored instead of listing a parent e.g. $\mathcal{T}_2 = [2,3,*]$. We can shift the tree by shifting all indices by a constant amount. This allows two disjoint trees to be combined straightforwardly with one tree occupying the first M_1 nodes and the remaining $M_2 = M - M_1$ nodes contain a shifted version of a tree on M_2 nodes e.g. $[\mathcal{T}_1, \mathcal{T}_2] = [2,3,1,5,6,*]$.

We will need to define several family relations later for encoding and decoding information within the parity tree. The *ancestors* of a tree node are the collection of all parents of parents of the node. We use the term *children* to refer to immediate children of the node. The *cousins* set refers to the set of all children of ancestors. First cousins, second cousins, and all higher order cousins are included in this definition. The younger cousins for a node are the cousins with index less than the node under consideration. We also need to define the younger *disjoint roots* for a particular node. This is the set of nodes without a parent that have index less than the node under consideration. One could generate several graphic representations of the parity trees as illustrated in our work on the Hubbard model.

II. FENWICK TREES

Fenwick trees [**?**] are a specific type of parity trees introduced in 1994 for the purpose of arithmetic coding. Earlier applications to binomial heaps appeared in the 1970's. They were rediscovered in 2000 by Bravyi and Kitaev in the context of quantum simulation. Recent work has further analyzed and simplified the the Bravyi-Kitaev presentation including various expositions (esp. those by Peter Love and co-workers) and its recent use in an experimental implementation by the Google quantum group. In this section, the connection that the Fenwick tree offers between the Jordan-Wigner and the Bravyi-Kitaev encodings will be elucidated. We will also cover the segmented Bravyi-Kitaev.

Let us begin with a definition of the Fenwick tree V. Havlíček suggested in our previous paper.

```
FENWICK( L , R )

   IF L==R       #done
   THEN RETURN

   #find middle
   M=FLOOR(L/2 + R/2)

   #make M a child of R
   CONNECT M TO R

   #recurse
   FENWICK(L   , M)
   FENWICK(M+1 , R)

END
```

One then generates a Fenwick tree over M nodes with FENWICK$(0, M-1)$.

One can also construct parity trees using the composition of several Fenwick subtrees. For example, we can partition the M fermionic modes into some other number of Fenwick tree of varying depth. For example, the $M = 8$ case could be described by a single Fenwick tree of depth 3

$$\mathcal{T}_3 = [2,4,4,8,6,8,8,*] \tag{4}$$

or two Fenwick trees of depth 2 $[2,4,4,*,6,8,8,*]$ or four Fenwick trees $[2,*,4,*,6,*,8,*]$ or finally, eight depth zero trees $[*,*,*,*,*,*,*,*]$.

The Bravyi-Kitaev transform corresponds to the use of a single Fenwick tree to store all nodes and the Jordan-Wigner consists of M trivial Fenwick trees each with one node. The correspondence between these two methods is not always clearly delineated and the use of the parity tree aims to clarify the connection between these transforms as well as other related transforms.

III. SPIN-FERMION TRANSFORMS USING PARITY TREES

By storing partial sums occasionally, the computation of prefix sums can be performed by querying less tree nodes. However, this increases the cost of extracting and updating the original occupancy bit string $K = k_1 k_2 ... k_M$. This section begins with a discussion of fermion and Majorana operators, followed by the connection to spin matrices. Finally, we consider the spin representation of fermion and Majorana operators.

The fermionic algebra is defined as $[a_j, a_k^\dagger]_+ = a_j a_k^\dagger + a_k^\dagger a_j = \delta_{jk}$ and $[a_j, a_k]_+ = 0$ for the set of operators corresponding fermionic modes $\{a_j\}$ and vacuum Ω with $a_k|\Omega\rangle = 0\ \forall k$. The action on the occupation basis, as dictated by the algebra in eq. (3), requires the flexibility to change both the occupancy and the phase of the state according the the parity of a prefix sum.

The Majorana operators are the symmetric and antisymmetric combinations of the creation and annihilation operators. The corresponding bosonic combinations are the $\hat{X}$ and $\hat{P}$ quadratures of the harmonic oscillator. The definition for Majorana operators we'll use is

$$c_j = (a_j + a_j^\dagger) \tag{5}$$

$$d_j = -i(a_j - a_j^\dagger) \tag{6}$$

All properties of the Majorana operators are derived from the fermionic algebra. Some important relations, with $x = c, d$, are

$$x_j^2 = \mathbf{1} \qquad a_j^\dagger a_j = \frac{1}{2}(\mathbf{1} + i c_j d_j)$$

$$x_i x_j = -x_j x_i \qquad a_k^\dagger a_j + a_j^\dagger a_k = \frac{i}{2}(c_k d_j + c_j d_k)$$

$$x_i = x_i^\dagger \qquad a_k^\dagger a_j^\dagger + a_j a_k = \frac{i}{2}(c_k d_j - c_j d_k)$$

Unlike the a_j and $a_j^\dagger$ operators, the c_j and d_j operators never destroy the basis states. The Majorana operators are unitary and Hermitian like the Pauli matrices.

We can represent the spin-$\frac{1}{2}$ algebra with the spin operators (which can further be represented by Pauli matrices) $\mathbf{1} = P_{00} + P_{11}$, $X = P_{10} + P_{01}$, $Y = i(P_{10} - P_{01})$, $Z = P_{00} - P_{11}$. One could have used the basis $P_{ij} = |i\rangle\langle j|$ for $i, j = 0, 1$, however $P_{10} = S_+ = \frac{1}{2}(X + iY)$ and $P_{01} = P_{01}^\dagger$ are not Hermitian and are identified with the raising and lowering operators.

$$P_{00} = \frac{\mathbf{1} + Z}{2} \tag{7}$$

$$P_{11} = \frac{\mathbf{1} - Z}{2} \tag{8}$$

$$P_{01} = S^+ = \frac{1}{2}(X + iY) \tag{9}$$

$$P_{10} = S^- = \frac{1}{2}(X - iY) \tag{10}$$

Let us assume that qubit states $|0\rangle$ and $|1\rangle$ are in the Z computational basis. Specifically, for a qubit string $|B\rangle = |b_1 b_2 ... b_M\rangle$. Then the Pauli matrices acts as $\mathbf{1}|b_j\rangle = |b_j\rangle$, $Z|b_j\rangle = (-1)^{b_j}|b_j\rangle$, $X|b_j\rangle = |\neg b_j\rangle$ and $Y|b_j\rangle = i(-1)^{\neg b_j}|\neg b_j\rangle$. Since the Pauli basis is unitary, we can simplify some of the spin operator expressions using the Majorana operators. Let us begin with the task of implementing the Majorana action on qubit strings in the parity tree basis: $c_j|T\rangle = (-1)^{\Gamma_j}|T'_{k'_j = \neg k_j}\rangle$.

To obtain the correct phase factor for nodes within the same tree, we need to pull the phase from the children of the node and from the younger cousins. The parity stored in the younger cousins of j will contain the modulus two sum of all lower indexed nodes within the tree. For the unrelated tree nodes the roots of disjoint trees is summed in the expression for Γ_j. Since $Z|t_j\rangle = (-1)^{t_j}|t_j\rangle$, we just apply to state $|T\rangle$ a tensor product of Z operators at each of the children, younger cousins and disjoint tree roots of node j i.e.

$$\hat{\Gamma}_j|T\rangle = Z_{Children} Z_{DisjointRoots} Z_{YCousins}|T\rangle = (-1)^{\Gamma_j}|T\rangle \tag{11}$$

If the occupancy k_j changes, t_j as well as the ancestors of j must be negated since they contain bit k_j in their summations. Hence, the full negation and phase computation is given by

$$c_j = \hat{\Gamma}_j \otimes X_j \otimes X_A \tag{12}$$

with the negation operator $X_A = X_{Ancestors(j)}$ applies a local X operator to the ancestors of j. As a small example, consider k_6 and Γ_6 of tree $\mathcal{T}_3$. For $\mathcal{T}_3$, $YCousins(6) = \{4\}$, $Children(6) = \{5\}$, $DisjointRoots(6) = \{\}$, $Ancestors(6) = \{7, 8\}$, thus:

$$k_6 = t_6 + t_5 \tag{13}$$

$$\Gamma_6 = k_5 + (k_4 + k_3 + k_2 + k_1) = t_5 + t_4 \tag{14}$$

$$c_6 = Z_4 Z_5 X_6 X_7 X_8 \tag{15}$$

Next, we want to consider the action of d_j. Observe, in the occupation basis,

$$\begin{aligned} d_j|K\rangle &= -i(a_j - a_j^\dagger) \prod_{m=1}^{M} (a_m^{k_m})^\dagger |\Omega\rangle \\ &= -i \left(\prod_{m=1}^{j-1} (-a_m^{k_m})^\dagger \right) \left[(a_j - a_j^\dagger)(a_j^{k_j})^\dagger \right] \left(\prod_{m=j+1}^{M} (a_m^{k_m})^\dagger \right) |\Omega\rangle \\ &= -i \left[\delta_{k_j 0}(0 - 1) + \delta_{k_j 1}(1 - 0) \right] (-1)^{\Gamma_j} |K'_{k'_j = \neg k_j}\rangle \\ &= i(-1)^{\neg k_j + \Gamma_j} |K'_{k'_j = \neg k_j}\rangle. \end{aligned} \tag{16}$$

In terms of the parity tree, we need $d_j|T\rangle = (-1)^{\neg k_j + \Gamma_j}|T'_{k'_j = \neg k_j}\rangle$.

The phase factor now depends on the occupancy k_j which requires knowing the occupancy of the site in addition to the prefix sum computation. Parity tree node t_j contains the binary sum of k_j and the sum of all the children t_{child} of tree node j. The updating of the ancestors using X_A and the computation of $(-1)^{\Gamma_j}$ using $\hat{\Gamma}_j$ are the same as the previous case. The key difference comes from replacing X_j with Y_j. Then,

$$\begin{aligned} Y_j|t_j\rangle &= i(-1)^{\neg t_j}|\neg t_j\rangle \\ &= i(-1)^{1 + t_j}|\neg t_j\rangle \\ &= i(-1)^{1 + k_j + \sum t_{child}}|\neg t_j\rangle \\ &= i(-1)^{1 + k_j}|\neg t_j\rangle \end{aligned} \tag{17}$$

Thus, the Y_j operator contributes a factor of $(-1)^{\sum t_{child}}$ which was already included in the expression for $\hat{\Gamma}_j$.

Hence, the phase factor computation for the children cancels out.

In summary,

$$c_j = \hat{\Gamma}_j \otimes X_j \otimes X_A \tag{18}$$

$$d_j = Z_{Children}\hat{\Gamma}_j \otimes Y_j \otimes X_A \tag{19}$$

The creation/annihilator operators are then $\frac{1}{2}(c_j \pm id_j)$ with + corresponding to annihilation.

IV. FURTHER READING

- P. M. Fenwick (1994). "A new data structure for cumulative frequency tables" *Software: Practice and Experience.* **24** (3): 327?336.
- S. Bravyi, A. Kitaev (2002). "Fermionic quantum computation" *Annals of Physics*, **298**, 210 (2002), quant-ph/0003137v2.
- M.-H. Yung, J. D. Whitfield, S. Boixo, D. G. Tempel, A. Aspuru-Guzik (2014). "Introduction to Quantum Algorithms for Physics and Chemistry" *Advances in Chemical Physics* Volume 154 (ed S. Kais), John Wiley & Sons.
- J. T. Seeley, M. J. Richard, P. J. Love (2012). "The Bravyi-Kitaev transformation for quantum computation of electronic structure." *Journal of Chemical Physics* **137** 22:224109.
- V. Havlíček, M. Troyer, J. D. Whitfield (2017). "Operator Locality in Quantum Simulation of Fermionic Models", arXiv:1701.07072v1 [quant-ph]