Lecture 8: Parity basis and Bravyi-Kitaev transform
Reading: Seeley, Richard, and Love JCP (2012)

1 Parity basis

The JW transform successfully transformed fermions to qubits, but it certainly has some drawbacks, namely that it turns local operators into nonlocal strings that result in many entangling gates being required to implement a given term. We would like to explore other encodings that may not have this undesirable property. The parity basis is an alternate encoding that will solve the issue with the JW transform at the expense of creating another nonlocal string associated with creation and annihilation operators. But we will see that it is a step on the way to the Bravyi-Kitaev transform which does require asymptotically more local strings than JW or the parity transform.

Recall that the problem with the JW transform was that we needed to figure out the parity of a state (to count minus signs) which then required us to determine the occupancy of a whole string of states. Wouldn't it be nice if we could somehow store the parity in a more local way? Then getting parity for an operator would only require a local operation.

This is precisely the idea of the parity basis, which stores the parity of all orbitals up to orbital j, e.g. orbital j stores $p_j = \sum_{s=0}^{j} f_s$ modulo 2. We can define a map between the parity and occupancy bases as:

$$p_i = \sum_j [\Pi_n]_{ij} f_j \tag{1.1}$$

where $[\Pi_n]_{ij} = 1$, $i \leq j$, and 0 otherwise. In other words, it is an upper diagonal matrix (including in the diagonal) filled with ones, and zero everywhere else (note the indices of the matrix are references from the bottom right of the matrix with i increasing horizontally from right to left, and j increasing vertically to be consistent with the orbital numbering from right to left in the ket).

Now, getting phase is easy because each qubit holds the parity, and parity determines the phase factor! We can get parity relevant for qubit j simply by applying Z to qubit $j-1$. The problem now is that adding or removing particles from the states is hard since each qubit does not store occupation. As an example if qubit $j-1$ is $|0\rangle$, then the creation operator is unchanged. But, if qubit $j-1$ is $|1\rangle$, then j needs inverted parity to represent an occupied state (since $1+1=0$ in mod 2 arithmetic). So a creation operator is now Q^- rather than Q^+. So now we have a two qubit operator on j and $j-1$:

$$\begin{aligned} P_j^{\pm} &\equiv Q^{\pm})j \otimes |0\rangle\langle 0|_{j-1} - Q_j^{\mp} \otimes |1\rangle\langle 1|_{j-1} & (1.2)\\ &= \frac{1}{2} X_j \otimes Z_{j-1} \mp iY_j \otimes I_{j-1} & (1.3)\end{aligned}$$

But that's not the only problem. Remember that the qubits at sites $> j$ store the parity for the other qubits. Those all need to be updated, meaning we once again have a non-local string. It is just like a JW string except it flips all the qubits, so it is an X string. The creation and annihilation operators in the parity basis are then:

$$\begin{aligned} a_j^{+} &\equiv \overleftarrow{X}_{j+1} \otimes P_j^{+} & (1.4)\\ a_j &\equiv \overleftarrow{X}_{j+1} \otimes P_j^{-} & (1.5)\end{aligned}$$

where $\overleftarrow{X}_i \equiv \sigma^x_{n-1} \otimes ... \otimes \sigma^x_i$. So we still have a nonlocal string.

2 Bravyi-Kitaev (BK) transform

2.1 Introduction

The BK basis is a middle ground between the JW and parity bases. Generally, we need two pieces of information to simulate fermionic operators with qubits: occupation of the target qubit and parity of orbitals with index less than the target index. In the occupation basis, occupation is local but parity is non-local. In the parity basis, occupation is nonlocal but parity is local.

The BK basis lets both the occupation and parity be nonlocal but requires strings of length $O(\log N)$ rather $O(N)$ as in the parity and occupation bases. It works in the following way. Even j qubits store occupation, while odd j qubits store the parity of a certain set of adjacent set of orbitals with index $< j$.

We can again define a map β_n that takes the representation from the occupation number basis to the BK basis: $b_i = [\beta_n]_{ij} f_j$. The map can be defined recursively in the following way. Let $\beta_1 = [1]$. Then

$$\beta_{2^{x+1}} = \begin{bmatrix} & 1 & 1 & ... \\ \beta_{2^x} & 0 & 0 & ... \\ & 0 & 0 & ... \\ 0 & & \beta_{2^x} & \end{bmatrix} \tag{2.1}$$

With this construction, we have a balance between the non-locality of the other bases. The parity of indices $< j$ are stored in a few partial sums, the number of which $\sim O(\log j) \leq O(\log n)$.

2.2 Representing operators

The hard part is to determine fermionic operators in the BK basis. We are not going to put in all the details here - they are presented fairly clearly in Seeley et al referenced at the top, although with a few errors that we point out here. Here is a sketch of the main considerations. There are 3 sets of qubits that we need to construct operators:

1. Parity set: those qubits that store the parity of all orbitals $< j$
2. Update set: those qubits that must be updated when the occupation of orbital j changes
3. Flip set: whether qubit j has the same parity as orbital j (and thus whether we should use a creation or annihilation operator on the qubit to represent adding a particle to the orbital)

Let's start with the parity set. For arbitrary j, which set of qubits tell us about the phase factor for an operator on orbital j? The parity of this qubit set gives us the parity of all orbitals with index $< j$, and so we define $P(j)$ as the parity set. We can figure out this set by mapping the BK basis to the parity basis using the two maps we already wrote down: we can first go from the BK basis to the occupation basis using β^{-1}, then from occupation to parity using Π' (defined below). The overall map is:

$$p_i = \sum_k [\Pi'_n \beta_n^{-1}]_{ik} b_k \tag{2.2}$$

Here Π'_n is the parity matrix defined earlier but with zeros on the diagonal since we need only the parity of orbitals less than j. This transform is in ERROR in the Seeley paper, which uses the same matrix as is used for the parity basis. The nonzero entries to the right of the diagonal indicate which qubits are needed to compute the cumulative parity to j. I give an example in class which is also given in the Seeley paper.

Next, consider the update set $U(j)$, defined to be the set of qubits other than j that must be updated when the occupation of orbital j changes. This set is therefore also the set of qubits in the BK basis that store a partial sum including orbital j. Any such qubit is defined to be in the set $U(j)$. Since even indexed qubits store only the occupation of orbital j, the update set only includes odd indices.

We can figure out this set using the map from the occupation number basis to the BK basis, which is just β. Since $b_i = \sum_{ij} [\beta_n]_{ij} f_j$, the columns of this matrix indicate which BK qubits store a particular orbital. More precisely, non-zero entries of a column above the diagonal tell us which qubits other than j must be updated if the occupation of j changes.

Finally, we have the flip set $F(j)$, which contains the qubits that store the parity of occupation numbers other than f_j in b_j. We need this information to tell us if qubit j has the same or inverted parity with respect to orbital j. As before, since even-indexed qubits only store the orbital with the same index, the flip set does not contain qubits with even indices, e.g. we know qubit j has the same parity as orbital j. We can get the flip set by using the map from the BK basis to the occupation number basis, $f_j = \sum_k [\beta_n^{-1}]_{jk} b_k$. For row j, the non-zeros entries to the right of the diagonal tell us which qubits other than j determine the occupation of qubit j.

With these sets defined, we now need to map the creation and annihilation operators to the BK basis. Unlike in the occupation number and parity bases where we focused on the properties of individual qubit states, for the BK transform we focus on the parity of subsets of orbitals and qubits. To start, define even and odd projectors for parity as

$$E_S = \frac{1}{2}(I + Z_S) \tag{2.3}$$

$$O_S = \frac{1}{2}(I - Z_S) \tag{2.4}$$

where Z_S is short-hand for applying the Z gate to all qubits in an arbitrary set S. Let's look at the case where j is even. $Q^{\pm}$ should act on qubit j, as in JW, since even qubits store orbital j. We then have to determine the parity of occupied orbitals $< j$ and update qubits with index $> j$. To figure out parity, we act with σ^z on all qubits $\in P(j)$ giving $Z_{P(j)}$. The size of this set is $O(\log j)$.

Next, we need to update the Update set: since j is now occupied, we need to update the partial sums containing f_j. We do that by applying σ^x to qubits in $U(j)$ giving $X_{U(j)}$. In the end, we have

$$a_j^\dagger = X_{U(j)} \otimes Q_j^+ \otimes Z_{P(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} - iX_{U(j)} \otimes Y_j \otimes Z_{P(j)}) \tag{2.5}$$

$$a_j = X_{U(j)} \otimes Q_j^- \otimes Z_{P(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} + iX_{U(j)} \otimes Y_j \otimes Z_{P(j)}) \tag{2.6}$$

These are the operators for j even. Now let's consider j odd. In this case, qubit j stores the partial sum of occupation numbers, not the occupation. The state of qubit j is representing either the occupation (if the parity of other orbitals $< j$ are 0) or the opposite of occupation (if the parity of other orbitals is 1). Therefore, which Q^+ or Q^- we need to use for creation or annihilation depends on the parity of the flip set, which we defined above. If the parity of $F(j) = 0$, we have the normal creation and annihilation operators, e.g. Q^+ creates a particle. If the parity of $F(j) = 1$, then we have the opposite: Q^- is the creation operator. Therefore, we can define an intermediate creation/annihilation operator as follows:

$$\Pi_j^{\pm} = Q_j^{\pm} \otimes E_{F(j)} - Q_j^{\mp} \otimes O_{F(j)} = \frac{1}{2} X_j \otimes Z_{F(j)} \mp iY_j \tag{2.7}$$

where the minus sign is accounting for the fact that parity is 1 by definition for the odd parity flip set and therefore we need the minus sign for the phase factor.

Now, there are still generally qubits in the parity set $P(j)$ that are not in the flip set $F(j)$. To get the final parity of the operator we need only those in $P(j)$ but not in $F(j)$ since we already accounted for the phase factor above. Therefore, we define a remainder set $R(j) \equiv P(j) \setminus F(j)$ to which the parity operator Z should be applied. The final creation and annihilation operators for j odd are then:

$$\begin{aligned} a_j^{\dagger} &= X_{U(j)} \otimes \Pi_j^{+} \otimes Z_{R(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} - iX_{U(j)} \otimes Y_j \otimes Z_{R(j)}) && (2.8) \\ a_j &= X_{U(j)} \otimes \Pi_j^{-} \otimes Z_{R(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} + iX_{U(j)} \otimes Y_j \otimes Z_{R(j)}) && (2.9) \end{aligned}$$

That Z is applied to $P(j)$ in the first terms is because the even parity projection operator has a $Z_{F(j)}$, so when we also apply $Z_{R(j)}$ we get $Z_{P(j)}$ as F and R by definition compose P.

If we compare the even and odd cases, the only difference is to which qubits Z is applied in the last Z gates. If we define $\rho(j) = P(j)$ for j even and $R(j)$ for j odd, we get the final result for all j:

$$\begin{aligned} a_j^{\dagger} &= X_{U(j)} \otimes \Pi_j^{+} \otimes Z_{R(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} - iX_{U(j)} \otimes Y_j \otimes Z_{\rho(j)}) && (2.10) \\ a_j &= X_{U(j)} \otimes \Pi_j^{-} \otimes Z_{R(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} + iX_{U(j)} \otimes Y_j \otimes Z_{\rho(j)}) && (2.11) \end{aligned}$$

2.3 Representing Hamiltonian terms

With the operators defined, it is straightforward but tedious to figure out how the relevant Hamiltonian terms are translated to Pauli gates in the BK representation. All the details are presented in Seeley, so I will just give one example. Let's consider the number operator $a_i^{\dagger} a_i$. From the definitions above:

$$\begin{aligned} a_i^{\dagger} a_i &= \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} - iX_{U(j)} \otimes Y_j \otimes Z_{\rho(j)}) \times && (2.12) \\ &\quad \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} + iX_{U(j)} \otimes Y_j \otimes Z_{\rho(j)}) \\ &= \frac{1}{4}(I + i(X_i Y_i) \otimes Z_{P(i)\setminus\rho(i)} - i(Y_i X_i) \otimes Z_{P(i)\setminus\rho(i)} + I) && (2.13) \\ &= \frac{1}{2}(I - Z_i \otimes Z_{P(i)\setminus\rho(i)}) && (2.14) \end{aligned}$$

It gets progressively more messy from here. Thankfully, we can have a computer do this translation for us.

2.4 BK Hamiltonian for minimal basis H_2

Applying this transform to minimal basis H_2 in which there are four spin-orbitals, we get the following result which we will use later (from O'Malley et al PRX):

$$\begin{aligned} H &= f_0 I + f_1 Z_0 + f_2 Z_1 + f_3 Z_2 + f_1 Z_0 Z_1 + f_y Z_0 Z_2 + f_5 Z_1 Z_3 \\ &+ f_6 X_0 Z_1 X_2 + f_6 Y_0 Z_1 Y_2 + f_7 Z_0 Z_1 Z_2 + f_4 Z_0 Z_2 Z_3 + f_3 Z_1 Z_2 Z_3 \\ &+ f_6 X_0 Z_1 X_2 Z_3 + f_6 Y_0 Z_1 Y_2 Z_3 + f_7 Z_0 Z_1 Z_2 Z_3 \end{aligned} \tag{2.15}$$

You start to see how many terms there are, even for the simplest possible problem! This Hamiltonian can be simplified somewhat: note that H acts off diagonally only on qubits 0 and 2 (the X gates). The simulation starts in a product state (corresponding to the HF molecular orbital basis) and therefore qubits 1 and 3 are never flipped. These qubits can thus be removed from the problem, leading to the Hamiltonian often cited in the literature (qubit numbers have been relabeled):

$$H = g_0 I + g_1 Z_0 + g_2 Z_1 + g_3 Z_0 Z_1 + g_4 X_0 X_1 + g_5 Y_0 Y_1 \tag{2.16}$$

The constants g are computed classically and only depend on the bond length selected. You can see that this Hamiltonian is much simpler than the original one! We are now ready to apply some algorithm to find the ground state and other properties of this Hamiltonian.