FCI Code

Bo-Xiao Zheng

December 16, 2013

1 Introduction and Objective

Full CI is the exact diagonalization technique used in quantum chemistry. Our goal is to write a full CI code based on the vertorized direct CI method.[1] The direct CI method uses revised Dividson algorithm[2] to find the lowest eigenvalue and the corresponding CI eigenvector. During the optimization process, the Hamiltonian will not be stored explicitly in matrix form, only the FCI vector **x** and **Hx** are stored. The configurations will be coded as ordered combinations of α and β strings, with a number 0 or 1 in each bit, representing the occupation of one MO. These combinations are mapped to lexical indices in a compact form.[3] Explicit lists of nonzero elements in 1- and 2-electron operators will be constructed and kept in memory, to reduce the time of evaluating **Hx** directly from CI vector.

The general steps of the program are:

1. Read input: number of orbitals and electrons, magnetic quantum number, one- and two-electron integrals of MOs.

2. Build the lists VO, O, OO, OV, OOV, OOVV. In each list, eg. $OV(i,j;s) = r$ means the operator $a_j^\dagger a_i$ maps state s to state r (with lexical index).

3. Initial guess, allocate memory for FCI vector **x** and **Hx**.

4. Construct and store diagonal elements of Hamiltonian matrix H_{ii}.

5. evaluate **Hx**, by going applying the diagonal Hamiltonian and go over all the possible mappings.

6. Use revised Davidson algorighm to update normalized CI vector **x** and the energy.

7. Check convergence. If not converged, go to step 5.

2 Alogrithm Description

2.1 Lexical Indexing

The lexical index is introduced using graphic representation. However, explicitly building the graph is not necessary for FCI. Consider only one of α and β spins, with N electrons and n orbitals. We first order the orbitals in the ascending order of energy. Rather than using the string of creating operators, we use the n-bit string of the occupation number to represent the slater determinant. The lexical order makes the Slater determinants occupying the highest energy orbitals has largest indices. Given a n-bit string $\{b_k\}$, the formula for its index is

$$Add(\{b_k\}) = \sum_{k=0}^{n-1} b_k C_k^{occ(k)} + 1 \tag{1}$$

Where occ(k) means the total occupation number of in the 0 to k orbitals. With this formula, the occupation string maps to integers 1 to C_n^N. The "+1" comes because we will distinguish $\pm Add$ later, and we don't want to confuse with ± 0.

On of the advantage of this mapping is that we can effeciently find its inverse. Given the index Add, the steps of finding the occupation string are

1. Add-=1, Start from number ($i = N, j = n-1$).

2. if $Add \geq C_j^i$, then $b_j = 1$,$Add- = C_j^i$,$i- = 1$,$j- = 1$; else $b_j = 0$,$j- = 1$.

3. If $j \geq 0$, repeat step 2.

The lexical mapping is used to construct the lists of non-vanishing Hamiltonian matrix elements before iteration. It will not be used in the main iteration. There, only the indices (addresses) are used. After the calculation converges, one might want to see the coefficients of specific Slater determinant, then the inverse mapping is useful there.

2.2 List Construction

Five different types of lists are to be constructed: O(i),OV(i,j), OO(i,j), OOV(i,j,k) and OOVV(i,j,k,l). In each list, we store all the possible occupations that conform to the occupation of the given orbital(s). the index (address) in a given list is the lexical index of the occupation string where the specified bits are suppressed, while the number stored is the address of the occupation string in the original (complete) list.

As a convenient trick, we store the phase factor of the corresponding matrix element as the sign along with the address. The complete definitions and methods to determine the phases are listed below:

1. O(i): the phase associated with $a_i^\dagger a_i$, always +1.

2. OV(i,j) ($i \neq j$): the phase associated with $a_j^\dagger a_i$, +1 if the number of occupied orbitals between orbital i and j is even, otherwise -1.

3. OO(i,j) ($i > j$): associated with $a_i^\dagger a_j^\dagger a_j a_i$), always +1.

4. OOV(i,j,k) (i,j,k differ from each other): associated with $a_i^\dagger a_k^\dagger a_j a_i$, i.e. exciting electron in orbital j to orbital k. The phase factor is +1 if number of occupied orbitals between j,k is even.

5. OOVV(i,j,k,l) ($i > j, k > l$): associated with the operator $a_l^\dagger a_k^\dagger a_i a_j$, +1 if sum of occupied orbitals between k and i, and l and j is even, otherwise -1. If $j > k$ or $l > i$, the opposite sign.

Use OV(i, j) as an example to explain how to build lists:(loop order may not be the optimal)

for add=0 to $C_{n-2}^{N-1} - 1$
  string1=add_to_string(add, no. of elec. = N-1, no. of MO = n-2)
  for i = 0 to n-1
    for j = 0 to i-1
      //build OV(i,j)
      string2=string1.insert(O,V,i,j)
      phase=occ(i,j)
      OV(i,j)[add]=string_to_add(string2 * phase, no. of elec. = N. no. of MO = n)
      //build OV(j,i)
      string2=string1.insert(O,V,j,i)
      OV(j,i)[add]=stringtoadd(string2 * phase, no. of elec. = N. no. of MO = n)

2.3 Operator Action

The broad picture of acting the operators on the FCI vector is, to go over each type of list, find the associated matrix elements (one- and two-electron integrals) and the map the coefficients of CI vector to **Hx**. In the process, spin should be appropriately considered. We will first work with $S_z = 0$ systems, the lists are identical for α and β spins. The extension to open-shell system is straightforward, we just need to use two different lists and seperate the mapping process. The FCI vector is stored in memory as a two-dimensional array, with each index specified by one of the occupation strings. We analyze the detailed algorithm of operator action in this section.

2.3.1 One-Electron Operators

One-electron part is the simplest case, where we only need the OV type of lists. α and β spins can be seperated in this case. Since

$$\begin{aligned}\hat{H}_1^\sigma &= \sum_{ij} h_{ij} a_{i\sigma}^\dagger a_{j\sigma} \\ &= \sum_i h_{ii} a_{i\sigma}^\dagger a_{i\sigma} + \sum_{i\neq j} h_{ij} a_{i\sigma}^\dagger a_{j\sigma} \\ &= \sum_i h_{ii} a_{i\sigma}^\dagger a_{i\sigma} + \sum_{i>j} h_{ij} (a_{i\sigma}^\dagger a_{j\sigma} + a_{j\sigma}^\dagger a_{i\sigma})\end{aligned} \tag{2}$$

1. case i=j:

i loop over all orbitals
  I loop over length of O(i)
    I_1= I'th entry of O(i)
    J loop over all β strings
      $Y(I_1, J) += X(I_1, J) \times h_{ii}$
    J loop over all α strings
      $Y(J, I_1) += X(J, I_1) \times h_{ii}$

2. case $i > j$:

i,j loop over all orbital pairs with $j < i$
  I loop over length of OV(i,j)
    I_1=I'th entry of OV(i,j)
    S_1=sign associated with I_1
    I_2=I'th entry of OV(j,i)
    S_2=sign associated with I_2
    J loop over β strings
      $Y(I_2, J) += X(I_1, J) \times h_{ij} \times S_1$
      $Y(I_1, J) += X(I_2, J) \times h_{ij} \times S_2$
    J loop over α strings
      $Y(J, I_2) += X(J, I_1) \times h_{ij} \times S_1$
      $Y(J, I_1) += X(J, I_2) \times h_{ij} \times S_2$

2.3.2 Two-Electron Operators

In the case of two electrons, we seperate $\alpha - \alpha$, $\beta - \beta$ and $\alpha - \beta$ cases. We also calculate diagonal elements seperately for use in Davidson iterations.

$$\hat{H}_2 = \frac{1}{2} \sum_{\mu\nu} \sum_{ijkl} \langle ij|kl\rangle a_{i\mu}^\dagger a_{j\nu}^\dagger a_{l\nu} a_{k\nu} \tag{3}$$

The two-electron integrals $\langle ij|kl\rangle$ follow the original definition. The quantum chemical conterpart is $(ik\|jl)$. Two essentially different cases are: $\mu = \nu$ and $\mu \neq \nu$. First consider $\mu \neq \nu$ case. Using the notation $E_{ij}^\mu = a_{i\mu}^\dagger a_{j\mu}$, then

$$H^{\alpha\beta} + H^{\beta\alpha} = \frac{1}{2} \sum_{ijkl} \langle ij|kl\rangle (E_{ik}^\alpha E_{jl}^\beta + E_{ik}^\beta E_{jl}^\alpha) \tag{4}$$

The expression only contains two one-electron excitations, which means we will only use the lists O(i) and

OV(i,j). Thus, four different cases should be considered: (a) $i=k, j=l$; (b) $i\neq k, j=l$; (c) $i=k, j\neq l$; (d) $i\neq k, j\neq l$. Later we will see case (b) and (c) can be merged.

Case (a) gives the diagonal part, which only involve the O(i) lists.

$$\begin{aligned} & D^{\alpha\beta}+D^{\beta\alpha} \\ &= \frac{1}{2}\sum_{ij}\langle ij|ij\rangle(E^{\alpha}_{ii}E^{\beta}_{jj}+E^{\beta}_{ii}E^{\alpha}_{jj}) \\ &= \sum_{i}\langle ii|ii\rangle E^{\alpha}_{ii}E^{\beta}_{ii} \\ &+ \sum_{i>j}\langle ij|ij\rangle(E^{\alpha}_{ii}E^{\beta}_{jj}+E^{\alpha}_{jj}E^{\beta}_{ii}) \end{aligned} \tag{5}$$

Case (b) and (c) involves one O(i) list and one OV(i,j) list.

$$\begin{aligned} A^{\alpha\beta} &= \frac{1}{2}\sum_{(i\neq k),j}\langle ij|kj\rangle(E^{\alpha}_{ik}E^{\beta}_{jj}+E^{\beta}_{ik}E^{\alpha}_{jj}) \\ &+ \frac{1}{2}\sum_{(j\neq l),i}\langle ij|il\rangle(E^{\alpha}_{ii}E^{\beta}_{jl}+E^{\beta}_{ii}E^{\alpha}_{jl}) \\ &= \sum_{(i\neq k),j}\langle ij|kj\rangle(E^{\alpha}_{ik}E^{\beta}_{jj}+E^{\beta}_{ik}E^{\alpha}_{jj}) \\ &= \sum_{(i>k),j}\langle ij|kj\rangle(E^{\alpha}_{ik}E^{\beta}_{jj}+E^{\alpha}_{jj}E^{\beta}_{ik}+E^{\alpha}_{ki}E^{\beta}_{jj}+E^{\alpha}_{jj}E^{\beta}_{ki}) \\ &= \sum_{(i>k),j}\langle ij|kj\rangle[E^{\alpha}_{jj}(E^{\beta}_{ik}+E^{\beta}_{ki})+(E^{\alpha}_{ik}+E^{\alpha}_{ki})E^{\beta}_{jj}] \end{aligned} \tag{6}$$

Case (d) involves two OV(i,j) lists and can be further divided into two parts.

$$\begin{aligned} B^{\alpha\beta} &= \frac{1}{2}\sum_{i\neq k, j\neq l}\langle ij|kl\rangle(E^{\alpha}_{ik}E^{\beta}_{jl}+E^{\alpha}_{jl}E^{\beta}_{ik}) \\ &= \sum_{i>k, j>l}\langle ij|kl\rangle(E^{\alpha}_{ik}+E^{\alpha}_{ki})(E^{\beta}_{jl}+E^{\beta}_{lj}) \end{aligned} \tag{7}$$

Here we first used the symmetry between i,k and j,l, reducing the summation to $i>k, j>l$.

The loops for the $\alpha-\beta$ case can be divided to handle each term:

1. The first term in diagonal part:

i loop over all orbitals
  I loop over length of O(i)
    I_1=I's entry of O(i)
    J loop over O(i)
      I_2=J's entry of O(i)
      $Y(I_1,I_2)+=\langle ii|ii\rangle\times X(I_1,I_2)$

2. The second term in diagonal part:

i, j loop over orbital pairs with $j<i$
  I loop over length of O(i)
    I_1=I's entry of O(i)
    J loop over O(j)
      J_1=J's entry of O(j)
      $Y(I_1,J_1)+=\langle ij|ij\rangle\times X(I_1,J_1)$
      $Y(J_1,I_1)+=\langle ij|ij\rangle\times X(J_1,I_1)$

3. $A^{\alpha\beta}$ term:

j loop over orbitals
  J loop over length of O(j)
    J_1=J's entry of O(j)
    i, k loop over orbital pairs with $k<i$
      I loop over length of OV(i,k)
        I_1=I'th entry of OV(i,k)
        I_2=I'th entry of OV(k,i)
        S_1=sign associated to I_1
        S_2=sign associated to I_2
        $Y(J_1,I_2)+=X(J_1,I_1)S_1\times\langle ij|kj\rangle$
        $Y(J_1,I_1)+=X(J_1,I_2)S_2\times\langle ij|kj\rangle$
        $Y(I_2,J_1)+=X(I_1,J_1)S_1\times\langle ij|kj\rangle$
        $Y(I_1,J_1)+=X(I_2,J_1)S_2\times\langle ij|kj\rangle$

4. $B^{\alpha\beta}$ term:

$(i>k)$ loop over orbital pairs
  $(j>l)$ loop over orbital pairs
    I loop over length of OV(i,k)
      J loop over length of OV(j,l)
        I_1 =I'th entry of OV(i,k)
        I_2 =I'th entry of OV(k,i)
        J_1 =J'th entry of OV(j,l)
        I_2 =J'th entry of OV(l,j)
        S_1 =sign associated to I_1
        S_2 =sign associated to J_1
        $Y(I_2,J_2)+=X(I_1,J_1)S_1\times S_2\times\langle ij|kj\rangle$
        $Y(I_2,J_1)+=X(I_1,J_2)S_1\times S_2\times\langle ij|kj\rangle$
        $Y(I_1,J_2)+=X(I_2,J_1)S_1\times S_2\times\langle ij|kj\rangle$
        $Y(I_1,J_1)+=X(I_2,J_2)S_1\times S_2\times\langle ij|kj\rangle$

The $\mu=\nu$ case:

$$\hat{H}_2^{\mu\mu}=\frac{1}{2}\sum_{ijkl}\langle ij|kl\rangle a^{\dagger}_{i\mu}a^{\dagger}_{j\mu}a_{l\mu}a_{k\mu} \tag{8}$$

Diagonal part requires $i\neq j$ and $i=k, j=l$ or $i=l, j=k$. The compact expression is

$$D^{\mu\mu}=\sum_{i>j}(\langle ij|ij\rangle-\langle ij|ji\rangle)a^{\dagger}_{i\mu}a^{\dagger}_{j\mu}a_{j\mu}a_{i\mu} \tag{9}$$

The off-diagonal part can be classified according to number of different indices. If there are 3 different

indices, then one of these 4 situations holds: $i = k, i = l, j = k, j = l$, and other indices must differ. The expression for this part is

$$V_1^{\mu\mu} = \sum_{j>k, i\neq j, i\neq k} (\langle ij|ik\rangle - \langle ij|ki\rangle) \times (a_{i\mu}^\dagger a_{j\mu}^\dagger a_{k\mu} a_{i\mu} + a_{i\mu}^\dagger a_{k\mu}^\dagger a_{j\mu} a_{i\mu}) \tag{10}$$

The three-index part is associated with the lists OOV(i,j,k), with i, j, k different from each other.

The last part has 4 different indices. Thus all i, j, k, l are different.

$$V_2^{\mu\mu} = \sum_{\substack{(i>j)>(k>l) \\ i,j,k,l \text{ different}}} (\langle ij|kl\rangle - \langle ij|lk\rangle)(a_{i\mu}^\dagger a_{j\mu}^\dagger a_{l\mu} a_{k\mu} + a_{l\mu}^\dagger a_{k\mu}^\dagger a_{i\mu} a_{j\mu}) \tag{11}$$

The loops for the $\alpha - \alpha$ and $\beta - \beta$ cases are:

6. Diagonal terms, which involve the lists OO(i,j) $(i > j)$:

i, j loop over orbital pairs with $i > j$
  $V = \langle ij|ij\rangle - \langle ij|ji\rangle$
  I loop over the length of OO(i,j)
    I_1=I'th entry of OO(i,j)
    J loop over all β strings
      $Y(I_1, J) += X(I_1, J) \times V$
    J loop over all α strings
      $Y(J, I_1) += X(J, I_1) \times V$

7. Three-index terms with OOV(i,j,k) lists:

j, k loop over orbital pairs with $j > k$
  i loop over orbitals with $i \neq j, k$
    $V = \langle ij|ik\rangle - \langle ij|ki\rangle$
    I loop over length of OOV(i,j,k)
      I_1=I'th entry of OOV(i,j,k)
      I_2=I'th entry of OOV(i,k,j)
      S_1=sign associated to I_1
      S_2=sign associated to I_2
      J loop over all β strings
        $Y(I_2, J) += X(I_1, J)S_1 \times V$
        $Y(I_1, J) += X(I_2, J)S_2 \times V$
      J loop over all α strings
        $Y(J, I_2) += X(J, I_1)S_1 \times V$
        $Y(J, I_1) += X(J, I_2)S_2 \times V$

8. Four-index terms, associated with OOVV(i,j,k,l) $(i > j, k > l)$:

i, j loop over orbital pairs with $i > j$
  k, l loop over orbitals with $k > l, (ij) > (kl)$ and no repeated index
    $V = \langle ij|kl\rangle - \langle ij|lk\rangle$
    I loop over the length of OOVV(i,j,k,l)
      I_1=I'th entry of OOVV(i,j,k,l)
      I_2=I'th entry of OOVV(k,l,i,j)
      S_1=sign associated to I_1
      S_2=sign associated to I_2
      J loop over all β strings
        $Y(I_2, J) += X(I_1, J)S_1 \times V$
        $Y(I_1, J) += X(I_2, J)S_2 \times V$
      J loop over all α strings
        $Y(J, I_2) += X(J, I_1)S_1 \times V$
        $Y(J, I_1) += X(J, I_2)S_2 \times V$

2.4 Iteration Algorithm

Davidson method is used to calculate the smallest values of a large matrix without explicitly writing down the matrix. It's a subspace algorithm so convergence is guaranteed. FCI converges fast with Davidson algorithm. For extreme case where FCI fails to converge in a few iterations, we use "subspace collapse" technique to avoid memory overflow. (this has not been implemented yet).

We follow exactly the routine for Davidson algorithm in the original paper, therefore no need to put it here.

The data structure for the algorithm will be a class named DavidsonUpdator, which consists the vectors from former iterations and a member function to update the vector which will be called in the iterations.

2.5 Parallelization

We use shared memory parallelization, namely, OpenMP. It accelerates the construction of mapping lists and the evaluation of $\mathbf{y} = \mathbf{Hx}$ significantly. Also the loop structures are optimized so that the processors access continuous memory rather than random memory.

2.6 Open-Shell Calculation

Open-shell extension is straight-forward. The only thing to consider is that the loop order need to be revised for higher efficiency.

References

[1] Gian Luigi Bendazzoli and Stefano Evangelisti. A vector and parallel full configuration interac-

tion algorithm. *The Journal of Chemical Physics*, 98(4):3141–3150, 1993.

[2] Ernest R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *Journal of Computational Physics*, 17(1):87 – 94, 1975.

[3] W Duch. Graphical representation of salter determinants. *Journal of Physics A: Mathematical and General*, 18(17):3283, 1985.