CI: Computational Details

Friday 18th January, 2019

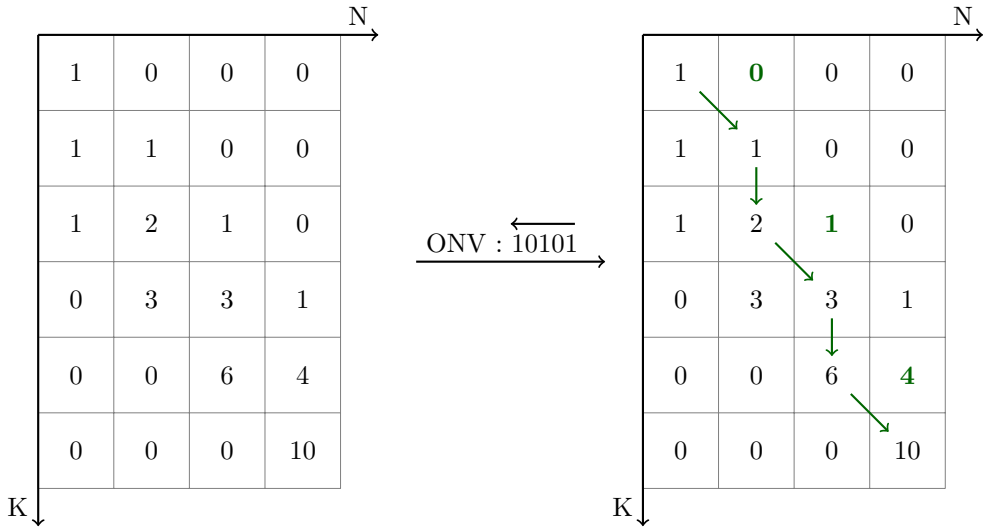
1 Two-Electron Off-Diagonal Address Calculation

1.1 Calculating Addresses

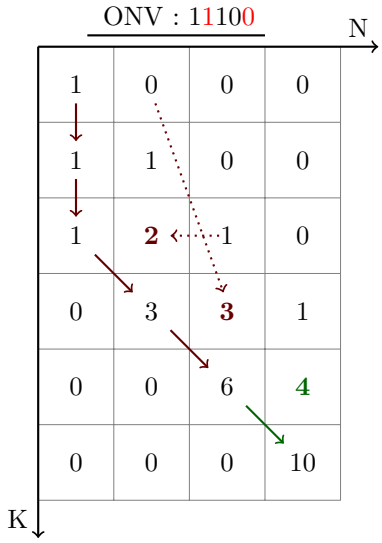
Calculating the address of an ONV can be done via the addressing scheme of the Fock space. (REF Lemmens 8.4.3 An addressing scheme for spin strings)
This is particullary important for DOCI where one cannot store sub calculations of the Hamiltonian, and the performance of the matvec will greatly be impacted by how fast you can generate the Hamiltonian (while in FCI one can perform the matvec with significantly smaller sparse matrixes).

1.1.1 One electron operators

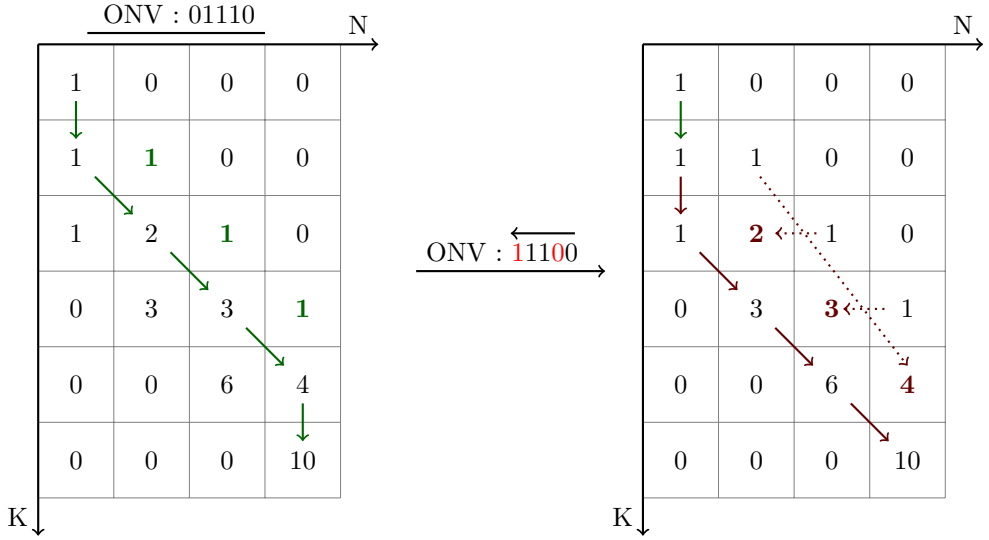
Take the following calculation on an ONV with address 5 (starting from 0) in a K=5 N=3 Fock space:



After performing an annihilation on electron position zero, and creating an electron on position 3 we arrivate at:



Only the path in-between the annihilation-creation couple is altered, hence we can calculate the new address from the old address. Subtract the weight of annihilated and add the weight of the created. Followed by subtracting the old weight of the altered path by that of the new weight to update the address. As an example, our initial address was 5, annihilated weight from electron one, on position zero: $W_{0,1}$, weight from the created electron (now electron two), on position three: $W_{3,2}$. The initial electron two was shifted to electron one (on position two): $W_{2,1} - W_{2,2}$ or $W_{3,2} - 2 * W_{2,2}$. This equates to $I - W_{0,1} + W_{3,2} - W_{2,2} + W_{2,1} = J \iff 5 - 0 + 3 - 1 + 2 = 9$
Below is an other example:



In this case two adjacent electrons are shifted. The total shift of multiple shifted electrons can be calculated simultaneously:

$$\sum_{i=0}^{l}(W_{k+i,n-1+i} - W_{k+i,n+i}) = W_{k+1+l,n-1+l} - W_{k+1+l,n+l} - W_{k,n-2} + W_{k,n-1} \tag{1}$$

starting from $W_{2,2}$ in the above example for two electrons ($i \in [0, 1]$):

$$W_{4,2} - W_{4,3} - W_{2,0} + W_{2,1} = 6 - 4 - 1 + 2 = 3 \tag{2}$$

With an initial address of 3, a creation-annhilation shift of 3 and a path shift of 3, we find $3 + 3 + 3 = 9$

It is often beneficial to generate all one-electron coupling sequentially so that your algorithm progresses sequentially through memory rather than discontionious or seemingly random.

Result: Generate all upper diagonal one-electron Hamiltonian elements
for *ONV in Fock space* **do**
  for *Occupied electron positions* **do**
    Set address and update according to annihilation
    if *next occupied position - initial position > 1* **then**
      update address through **shift**
      $sign = -1^{nextoccupiedposition - initialposition - 1}$
    end
    while *unoccupied positions left* **do**
      Calculate **new** address according to creation and calculate the Hamiltonian/RDM element according to the sign and the creation and annhilation indexes
      if *next occupied position - initial position > 1* **then**
        update address through **shift**
        $sign = sign * -1^{nextoccupiedposition - initialposition - 1}$
      end
    end
  end
end

Algorithm 1: Possible one-electron operator algorithm

The initial total operator sign will always be 1, because if no shift has occured over occupied electrons the creation operator will have the same sign as the annihilation operator making the total sign positive (-1*-1 = 1 or 1*1 = 1).

1.1.2 Two electron operators

For DOCI the only non-vanishing off-diagonal coupling operator combinations are two-electron operators which will affect both alpha in beta in the same way such that they can be consdensed to one-electron operators in one Fock space. One can then perform algorithm (1) but without the operator sign (it's always positive). Implementing these efficiently is very important for the DOCI matvec algoritm as its scaling will soley depend on how quickly you can generate all Hamiltonian elements (when one cannot store the Hamiltonian in a sparse matrix)

For FCI, the two-electron algorithm will be a combination of next-shift and back-shift calculations. And which operator combinations are minimal will be discussed in the following section. However, performance of the matvec will not be affected. Because the Hamiltonian elements can be stored sparsely virtually always (see later). The couplings and elements will have to be cacluated once, before the matvec, and will affect performance if it is written so poorly that it requires more execution time than a matvec, or than multiple matvec calls durning the Davidson algorithm.

1.2 Hamiltonian Elements

Given a Fock space.
Let us call $|I_\alpha\rangle$ the ONV with address I. We have arrived at following expression for the Hamiltonian (ref lemmens):

$$\hat{\mathcal{H}}_{\text{elec}} = \sum_{pq}^{K} k_{pq}\hat{E}_{pq} + \frac{1}{2}\sum_{pqrs}^{K} g_{pqrs}\hat{E}_{pq}\hat{E}_{rs} \tag{3}$$

We will focus only on the two electron operators $\sum_{pqrs}^{K} g_{pqrs}\hat{a}_p^\dagger\hat{a}_q\hat{a}_r^\dagger\hat{a}_s$. And only focus on the α electrons. We will therefore ignore the α subscript. We require to consider operator indices for which a given ONV I does not vanish:

$$\langle I|\,\hat{a}_p^\dagger\hat{a}_q\hat{a}_r^\dagger\hat{a}_s = \langle J| \tag{4}$$

In which J is an address larger than I. Reason for this is, in the event that $|I\rangle$ can be transformed in $|J\rangle$. $|J\rangle$ can also be transformed back to $|I\rangle$ yielding the same two-electron term (hermitian two-electron operators).

$$(\langle I|\,E_{pq}E_{rs}\,|J\rangle)^\dagger = \langle J|\,E_{sr}E_{qp}\,|I\rangle \tag{5}$$

It is important to note, that I will base further explanations from the perspective of equation (4) where $\hat{a}_p^\dagger$ annihilates on $\langle I|$.

1.2.1 Minimal operator iterations

For the address to be larger at all times, the highest index of a creation should always be higher than the highest index of an annihilation. This is easily verified by the fact that we represent our ONVs in binary and that the addressing is based on the ordering of its integer value. Given the relation of numeric value for each index of an integer represented in binary is quadratic, the integer value of a set index is always larger than any combination of previously set indices:

$$2^N - 1 = \sum_{i=0}^{N-1} 2^i \tag{6}$$

Additionally we can state that the first annihiltion can always have a smaller index than the second annihiltion without skipping over any address, the same is true for the creation operators.

Regardless in which way they are executed (if they are all different indices), the address will be the same. However the order of execution can alter the expression (sign wise) and will be accompanied by a different two-electron term. Given an ONV $\langle I|$ for which $\langle I|\,\hat{a}_p^\dagger\hat{a}_q\hat{a}_r^\dagger\hat{a}_s \neq 0$ we find that:

$$\begin{aligned}
\langle I|\,\hat{a}_p^\dagger\hat{a}_q\hat{a}_r^\dagger\hat{a}_s &= \langle I|\,(\hat{a}_p^\dagger\hat{a}_s\delta_{rq} - \hat{a}_p^\dagger\hat{a}_r^\dagger\hat{a}_q\hat{a}_s) \\
&= \langle I|\,(\hat{a}_p^\dagger\hat{a}_s\delta_{rq} + \hat{a}_r^\dagger\hat{a}_p^\dagger\hat{a}_q\hat{a}_s) \\
&= \langle I|\,(\hat{a}_p^\dagger\hat{a}_s\delta_{rq} + \hat{a}_r^\dagger\hat{a}_s\delta_{pq} - \hat{a}_r^\dagger\hat{a}_q\hat{a}_p^\dagger\hat{a}_s) && (7) \\
&\quad \text{IF p,q,r,s} \neq \\
&= -\langle I|\,(\hat{a}_r^\dagger\hat{a}_q\hat{a}_p^\dagger\hat{a}_s) && (8) \\
&= \langle I|\,(\hat{a}_r^\dagger\hat{a}_s\hat{a}_p^\dagger\hat{a}_q) && (9) \\
&= -\langle I|\,(\hat{a}_p^\dagger\hat{a}_s\hat{a}_r^\dagger\hat{a}_q) && (10)
\end{aligned}$$

So for the $p,q,r,s \neq$ case, we can enforce : $p < r$, $q < s$ for symmetries and anti-symmetries, and $s > r$ upper diagonal to not generate redundant addresses. This leaves us with limited combinations:

1. $p > q$ $(s > r > p)$
2. $p < q$
   - $r > q$ $(s > r)$
   - $q > r$ $(s > q)$

For inplace annihila-crea- and crea-annihila-tions, the rules are slightly different, because creation annihilation operators with the same index cancel each other out. Therefore the non-annihiltion bound creation index has to be larger than the non-creation bound annihiltion index to produce larger addresses (the rules for one-electron evaluation).

1. $p = q$, $s > r$
2. $q = r$, $s > p$

These have some implication for symmetry and anti-symmetry equations such as equation (7) as δ is not always zero:

$$\begin{aligned}
\langle I|\, \hat{a}_p^\dagger \hat{a}_q \hat{a}_r^\dagger \hat{a}_s &= \langle I|\, (\hat{a}_p^\dagger \hat{a}_s \delta_{rq} + \hat{a}_r^\dagger \hat{a}_s \delta_{pq} - \hat{a}_r^\dagger \hat{a}_q \hat{a}_p^\dagger \hat{a}_s) && (11)\\
&= \langle I|\, (\hat{a}_p^\dagger \hat{a}_s \delta_{rq} + \hat{a}_r^\dagger \hat{a}_s \delta_{pq} - \hat{a}_r^\dagger \hat{a}_q \delta_{sp} - \hat{a}_r^\dagger \hat{a}_s \delta_{pq} + \hat{a}_r^\dagger \hat{a}_s \hat{a}_p^\dagger \hat{a}_q) && (12)\\
&= \langle I|\, (\hat{a}_p^\dagger \hat{a}_q \delta_{rs} + \hat{a}_p^\dagger \hat{a}_s \delta_{qr} - \hat{a}_p^\dagger \hat{a}_s \hat{a}_r^\dagger \hat{a}_q) && (13)
\end{aligned}$$

1.2.2 $p = q$

For $p = q$ we can also see that for $s = q \iff r = q$ ortherwise we would have a double creation on the same index without an annihilation on that same index, which is a vanishing operation sequence. However this does not alter the address (diagonal contribution) and is ignored in the algorithm. Hence we state that $s \neq p, q, r$ This simplifies the equations:

$$\begin{aligned}
\langle I|\, \hat{a}_p^\dagger \hat{a}_p \hat{a}_r^\dagger \hat{a}_s &= \langle I|\, (\hat{a}_p^\dagger \hat{a}_s \delta_{rp} + \hat{a}_r^\dagger \hat{a}_s - \hat{a}_r^\dagger \hat{a}_p \hat{a}_p^\dagger \hat{a}_s) && (14)\\
&= \langle I|\, (\hat{a}_p^\dagger \hat{a}_s \delta_{rp} + \hat{a}_r^\dagger \hat{a}_s \hat{a}_p^\dagger \hat{a}_p) && (15)\\
&= \langle I|\, (\hat{a}_p^\dagger \hat{a}_s \delta_{pr} - \hat{a}_p^\dagger \hat{a}_s \hat{a}_r^\dagger \hat{a}_p) && (16)
\end{aligned}$$

We can then discriminate between $r = p$:

$$\begin{aligned}
\langle I|\, \hat{a}_p^\dagger \hat{a}_p \hat{a}_p^\dagger \hat{a}_s &= \langle I|\, (\hat{a}_p^\dagger \hat{a}_s + \hat{a}_p^\dagger \hat{a}_s - \hat{a}_p^\dagger \hat{a}_p \hat{a}_p^\dagger \hat{a}_s) && (17)\\
&= \langle I|\, (\hat{a}_p^\dagger \hat{a}_s + \hat{a}_p^\dagger \hat{a}_s \hat{a}_p^\dagger \hat{a}_p) && (18)\\
&= \langle I|\, (\hat{a}_p^\dagger \hat{a}_s - \hat{a}_p^\dagger \hat{a}_s \hat{a}_p^\dagger \hat{a}_p) && (19)
\end{aligned}$$

We see in equation (18) and (19) that last term annihilates p, then operator on index s does strictly not create on index p and index p is annihilated again, thus this term vanishes:

$$\langle I| \hat{a}_p^\dagger \hat{a}_p \hat{a}_p^\dagger \hat{a}_s = \langle I| \hat{a}_p^\dagger \hat{a}_s \tag{20}$$

For $r \neq p$:

$$\begin{aligned}
\langle I| \hat{a}_p^\dagger \hat{a}_p \hat{a}_r^\dagger \hat{a}_s &= \langle I| (\hat{a}_r^\dagger \hat{a}_s - \hat{a}_r^\dagger \hat{a}_p \hat{a}_p^\dagger \hat{a}_s) \\
&= \langle I| (\hat{a}_r^\dagger \hat{a}_s) && (21) \\
&= \langle I| (\hat{a}_r^\dagger \hat{a}_s \hat{a}_p^\dagger \hat{a}_p) && (22) \\
&= - \langle I| (\hat{a}_p^\dagger \hat{a}_s \hat{a}_r^\dagger \hat{a}_p) && (23)
\end{aligned}$$

Where se see that for equation (21) the second term vanished, as the initial term is assumend non-vanishing.

1.2.3 $q = r$

We only cover $p \neq r$ as we assume tha r starts unoccupied as opposed to the previous section (1.2.2).

$$\begin{aligned}
\langle I| \hat{a}_p^\dagger \hat{a}_r \hat{a}_r^\dagger \hat{a}_s &= \langle I| (\hat{a}_p^\dagger \hat{a}_s - \hat{a}_r^\dagger \hat{a}_r \hat{a}_p^\dagger \hat{a}_s) \\
&= \langle I| \hat{a}_p^\dagger \hat{a}_s && (24) \\
&= \langle I| (\hat{a}_p^\dagger \hat{a}_s + \hat{a}_r^\dagger \hat{a}_s \delta_{pr} - \hat{a}_r^\dagger \hat{a}_r \delta_{sp} - \hat{a}_r^\dagger \hat{a}_s \delta_{pr} + \hat{a}_r^\dagger \hat{a}_s \hat{a}_p^\dagger \hat{a}_r) && (25) \\
&= \langle I| (\hat{a}_p^\dagger \hat{a}_r \delta_{rs} + \hat{a}_p^\dagger \hat{a}_s - \hat{a}_p^\dagger \hat{a}_s \hat{a}_r^\dagger \hat{a}_r) && (26)
\end{aligned}$$

Which simplifies to:

$$\langle I| \hat{a}_p^\dagger \hat{a}_r \hat{a}_r^\dagger \hat{a}_s = \langle I| \hat{a}_p^\dagger \hat{a}_s \tag{27}$$

1.2.4 Summary for the Hamiltonian

In short this we shall sum up what appears to be the minimal amount of operators required to retrieve all information for the Hamiltonian for the two-electron (same spin) operators. The value:

$$\frac{1}{2}(g_{pqrs} + g_{rspq} - g_{rqps} - g_{psrq}) \tag{28}$$

Can be retrieved for any of the follwing non-vanishing operator sequence combinations yielding a higher and the same (thus non redundant) address for a given $\langle I|$:

1. $s > r > p > q$
2. $s > r > q > p$
3. $s > q > r > p$

For every occupied index x in an ONV:

$$\frac{1}{2}(g_{xxpq} + g_{pqxx} - g_{xpqx}) \tag{29}$$

We can also see that for $p = x$ we arrive at:

$$\frac{1}{2}(g_{xxxq}) \tag{30}$$

For every unnocupied index y ($p \neq y$) in an ONV:

$$\frac{1}{2}(g_{pyyq}) \tag{31}$$

Where for both cases $p < q$.

2 The FCI (restricted) matrix vector product

Internally it is recommended to work with relative addresses for both spin functions and then order them logically in your total vectors. Given the nature of our Hamiltonian (total operators do not affect spin) we can write an ONV as two ONVs from two seperate Fock spaces (and have the total Fock space be a direct product of those spaces). $|ONV\rangle = |ONV_\alpha ONV_\beta\rangle$. Choose one of the ONVs to be major and the other minor. If for example your α Fock space is major your ONV_α will permutate after a full permutation of your ONV_β which are β Fock space dimension permutations. We define $\mathbf{dim}_\alpha$ as the α Fock space dimension and $\mathbf{dim}_\beta$ as the β Fock space dimension and the total dimension $\mathbf{dim}_{total} = \mathbf{dim}_\alpha * \mathbf{dim}_\beta$.

An eigenvector is thus stored as such:

I_α	I_β	$I_{\mathbf{total}}$
I_1	I_1	I_1
I_1	I_2	I_2
I_1	I_3	I_3
I_1	$I_{...}$	$I_{...}$
I_1	$I_{\mathbf{dim}_\beta}$	$I_{\mathbf{dim}_\beta}$
I_2	I_1	$I_{\mathbf{dim}_\beta+1}$
I_2	I_2	$I_{\mathbf{dim}_\beta+2}$
$I_{...}$	$I_{...}$	$I_{...*\mathbf{dim}_\beta+...}$
$I_{\mathbf{dim}_\alpha}$	$I_{\mathbf{dim}_\beta}$	$I_{\mathbf{dim}_{total}}$

Rudimentary approach to performing the matrix vector product for a none storable Hamiltonian can be done as follows (where P is the vector resulting from the product and X is the vector partaking in the product):

$$\mathrm{P}_I = \sum_J \mathrm{H}_{IJ} * \mathrm{X}_J \tag{32}$$

$$\mathrm{P}_J = \sum_I \mathrm{H}_{JI} * \mathrm{X}_I \tag{33}$$

Where $\mathrm{Hamiltonian}_{JI} = \mathrm{Hamiltonian}_{IJ}$ (or $\mathrm{Hamiltonian}_{JI} = \mathrm{Hamiltonian}^*_{IJ}$ when working complex). Allowing for the upperdiagonal approach (for I coupling to J one finds J coupling to I)

2.1 Evaluations Seperated by Spin

For a set of operators only affecting one spin function we can repeat the coupling as many times as there are permutations in the Fock space of the opposite spin. e.g.:

$$\langle I_\alpha I_\beta | \, \hat{a}^\dagger_{p\alpha} \hat{a}_{q\alpha} \hat{a}^\dagger_{r\alpha} \hat{a}_{s\alpha} \, | J_\alpha I_\beta \rangle \neq 0 \tag{34}$$

We can calculate a portion of P (all one-electron and part of the two-electron (only the same spin) evaluations), P^{α} and P^{β}. Where we now define $\mathrm{P} = \mathrm{P}^{\alpha\beta} + \mathrm{P}^{\alpha} + \mathrm{P}^{\beta}$ with $\mathrm{P}^{\alpha\beta}$ exclusively operator combinations working on both spin functions simultaneously. Equation (34) holds for any I_β. Allowing to re-use calculated Hamiltonian elements related to the coupling.

$$\forall I_\beta : \mathrm{P}^{\alpha}_{I_\alpha * \mathbf{dim}_\beta + I_\beta} = \sum_{J_\alpha > I_\alpha} \mathrm{H}^{\alpha}_{I_\alpha J_\alpha} * \mathrm{X}_{J_\alpha * \mathbf{dim}_\beta + I_\beta} \tag{35}$$

$$\forall I_\beta : \mathrm{P}^{\alpha}_{J_\alpha * \mathbf{dim}_\beta + I_\beta} = \sum_{I_\alpha > J_\alpha} \mathrm{H}^{\alpha}_{J_\alpha I_\alpha} * \mathrm{X}_{I_\alpha * \mathbf{dim}_\beta + I_\beta} \tag{36}$$

Where the H^{α} refers to the Hamiltonian values retrievable by exclusively operators working on alpha electrons. Base-line implementation (with a for loop) will result significant difference in execution speed for both spin functions. As an example we have an FCI calculation with $K = 12$ molecular orbitals and 6 α electrons and 6 β electrons. Yielding an Alpha and Beta Fock space of dimenion 924 and a total dimension of 853.776k where alpha is major as in the vector example above. Calculating all couplings with Alpha and perfoming the matvec takes: $249ms$ versus Beta: $2561ms$. The alpha matvec is a lot cheaper because it gets repeated in an uninterrupted sequencel matter, versus beta which repeats with a large interval (the dimenion of the alpha Fock space) causing cache misses. This was solved using the Eigen3 (ref EIGEN) API for efficiently performing the matvec in a vectorized matter. (There is no motivation other than educational to attempt to write a library with similar functionallity and performance as Eigen3 by yourself)

The resulting vector and the vector from the product can both be mapped to a matrix representation.

$$\begin{bmatrix} I^{\alpha}_1 I^{\beta}_1 & I^{\alpha}_1 I^{\beta}_2 & I^{\alpha}_1 I^{\beta}_{\dots} & \dots & I^{\alpha}_1 I^{\beta}_{\mathbf{dim}_\beta} \\ I^{\alpha}_2 I^{\beta}_1 & I^{\alpha}_2 I^{\beta}_2 & I^{\alpha}_2 I^{\beta}_{\dots} & \dots & I^{\alpha}_2 I^{\beta}_{\mathbf{dim}_\beta} \\ I^{\alpha}_{\dots} I^{\beta}_1 & I^{\alpha}_{\dots} I^{\beta}_2 & I^{\alpha}_{\dots} I^{\beta}_{\dots} & \dots & I^{\alpha}_{\dots} I^{\beta}_{\mathbf{dim}_\beta} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I^{\alpha}_{\mathbf{dim}_\alpha} I^{\beta}_1 & I^{\alpha}_{\mathbf{dim}_\alpha} I^{\beta}_2 & I^{\alpha}_{\mathbf{dim}_\alpha} I^{\beta}_{\dots} & \dots & I^{\alpha}_{\mathbf{dim}_\alpha} I^{\beta}_{\mathbf{dim}_\beta} \end{bmatrix} \tag{37}$$

One can then calculate H^{α} and H^{β} which are $\mathbf{dim}_\alpha$ by $\mathbf{dim}_\alpha$ and $\mathbf{dim}_\beta$ by $\mathbf{dim}_\beta$ matrixes respectively. Note that these will be sparse and will only cost a fraction to store compared P and X. Let us call $\mathcal{P}$: P mapped as the matrix in eqation (37)

$$\forall I_\beta : \mathcal{P}^{\alpha}_{I_\alpha I_\beta} = \sum_{J_\alpha} \mathrm{H}^{\alpha}_{I_\alpha J_\alpha} * \mathbb{X}_{J_\alpha I_\beta} \tag{38}$$

$$\forall I_\alpha : \mathcal{P}^{\beta}_{I_\alpha I_\beta} = \sum_{J_\beta} \mathrm{H}^{\beta}_{I_\beta J_\beta} * \mathbb{X}_{I_\alpha J_\beta} \tag{39}$$

$$\mathcal{P}^{\alpha} = \mathrm{H}^{\alpha} * \mathbb{X} \tag{40}$$

$$\mathcal{P}^{\beta} = \mathbb{X} * \mathrm{H}^{\beta} \tag{41}$$

Eigen3 offers a sparse matrix module (initialize it with vector of `Eigen::Triplets`). One can either only fill the upperdiagonal and use `Eigen::SparseMatrix::selfadjointView<upper>()` when performing a matrix multiplication, however I have found that filling both upper and lower diagonal of the self adjoint `SparseMatrx` significantly reduces the execution time of the matrix multiplication.

2.2 Evaluations of the two-electron mixed spin operators

As found in Helgaker (ref) we find that we can calculate alpha-beta mixed elements by calculating all one-electron couplings for alpha in the alpha Fock space for a given annihilation-creation pair and storing these in a sparse matrix called $\sigma^{\alpha pq}$ (dim^2_α) with p and q being the annihilation and creation indexes respectively. We can then calculate all one-electron beta couplings for all operator combinations r,s and store two-electron integrals g_{pqrs} in the matrix according to the chosen alpha p, q couplings, we call this matrix : $\theta^{\beta pq}$ (dim^2_β).

$$\mathcal{P}^{\alpha\beta} = \sum_{pq} \mathcal{P}^{\alpha\beta pq} = \sum_{pq} \sigma^{\alpha pq} * \mathbb{X} * \theta^{\beta pq} \tag{42}$$

$$\sigma^{\alpha pq}_{I_\alpha J_\alpha} = \langle I_\alpha | \, E^\alpha_{pq} \, | J_\alpha \rangle \tag{43}$$

$$\theta^{\beta pq}_{I_\beta J_\beta} = \sum_{rs} g_{pqrs} \, \langle I_\beta | \, E^\beta_{rs} \, | J_\beta \rangle \tag{44}$$

Because $g_{pqrs} = g_{qprs}$ one can include both p, q and q, p in a single alpha coupling matrix, allowing $\sum_p \sum_{q \geq p}$. Additionally if you want to omit diagonal calculations, simply omit any calculations where $r = s$ from the $\theta^{\beta pq}$ if $p = q$.

2.3 Memory requirements

Between each matrix vector product we have some required storage (P and X, in our code also the diagonal) and some optional. Minimal storage (in arbitrary units) required is approximately: $3 * \dim_{total} = 3 * \binom{K}{N_\alpha}\binom{K}{N_\beta}$ For the optional intermediates we need:

- $\mathrm{H}^\alpha = \binom{K}{N_\alpha} * \left(\frac{\binom{K-N_\alpha}{2}}{2} * (N^2_\alpha - N_\alpha) + (K - N_\alpha) * N_\alpha\right)$
- $\mathrm{H}^\beta = \binom{K}{N_\beta} * \left(\frac{\binom{K-N_\beta}{2}}{2} * (N^2_\beta - N_\beta) + (K - N_\beta) * N_\beta\right)$
- $\sigma^{\alpha pq} * (K^2 - K) \approx \binom{K-1}{N_\alpha - 1} * (K^2 - K)$
- $\theta^{\beta pq} * (K^2) \approx \binom{K}{N_\beta} * (K - N_\beta) * N_\beta * K^2$

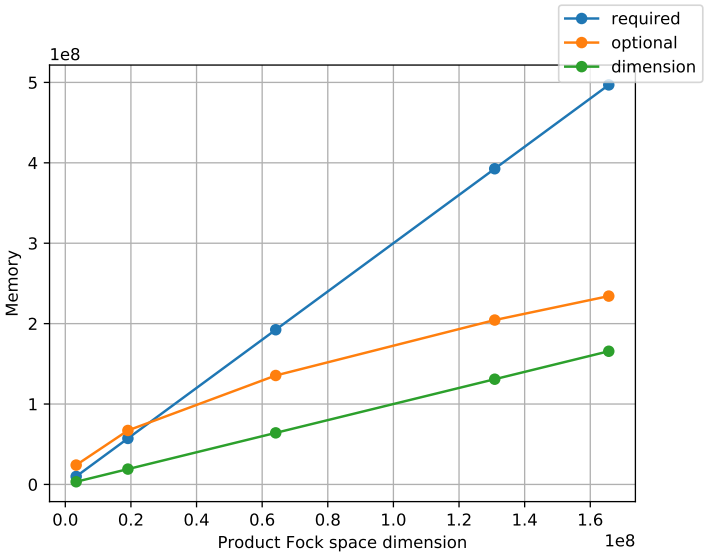


Figure 1: K=16 and $N_\alpha = N_\beta[4, 8[$

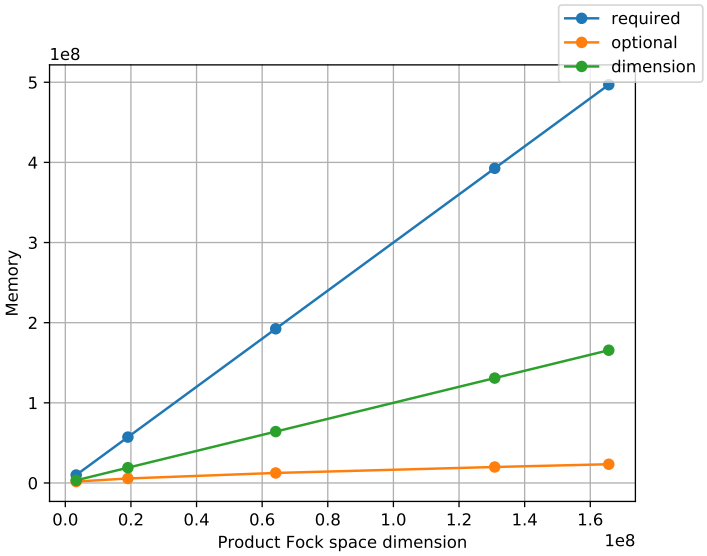


Figure 2: K=16 and $N_\alpha = N_\beta[4, 8[$ no θ

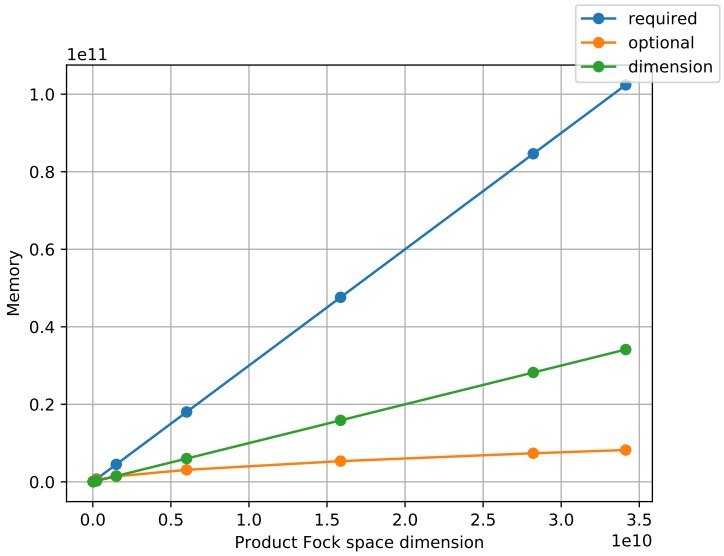


Figure 3: K=20 and $N_\alpha = N_\beta[4, 10[$

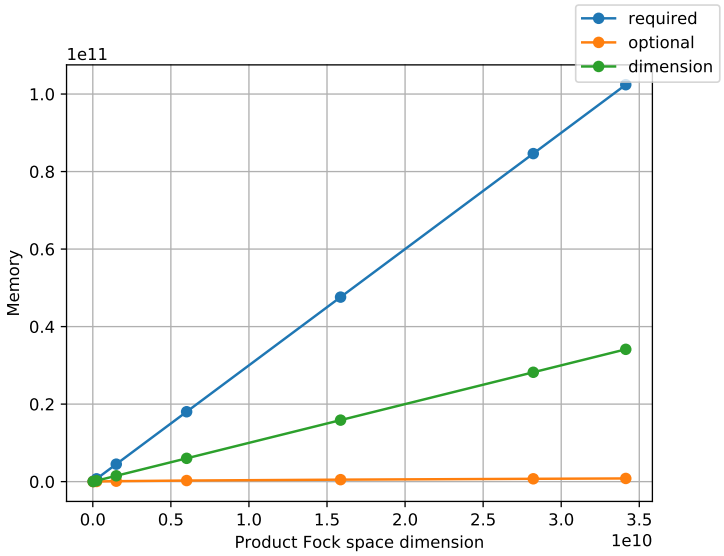


Figure 4: K=20 and $N_\alpha = N_\beta[4, 10[$ no θ

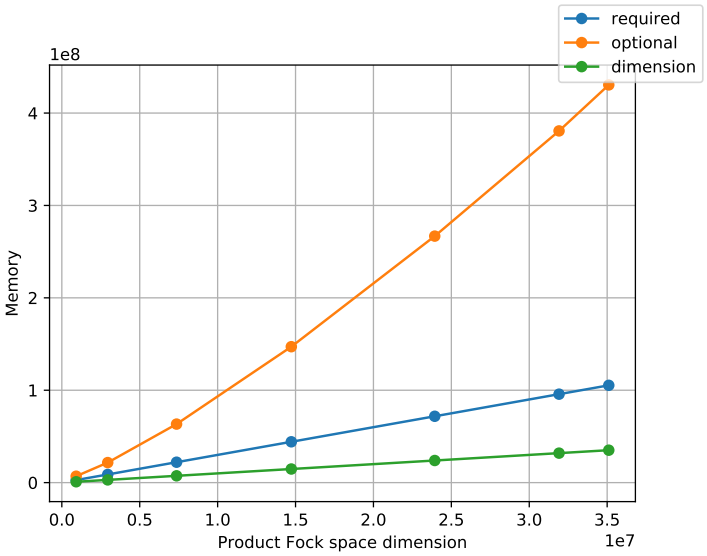


Figure 5: K=20 and $N_\beta = 2, N_\alpha[4, 10[$

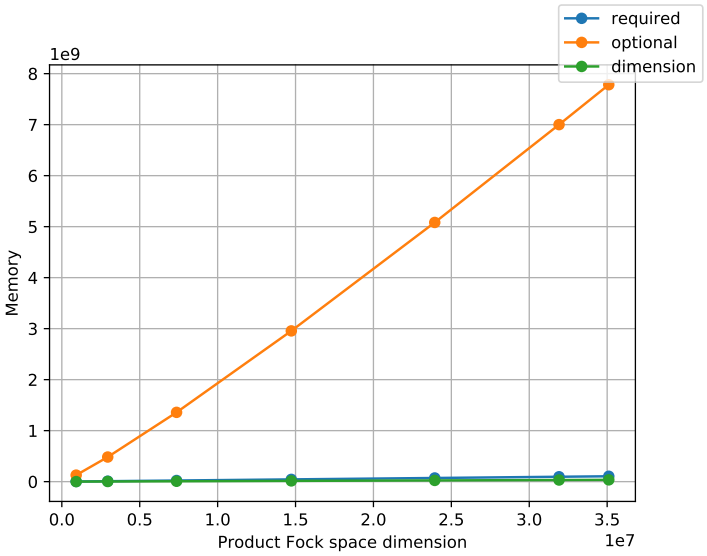


Figure 6: K=20 and $N_\alpha = 2, N_\beta[4, 10[$

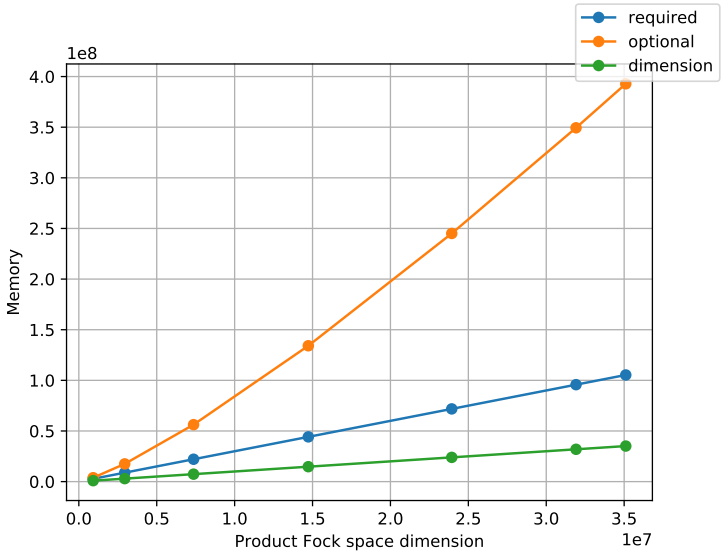


Figure 7: K=20 and $N_\alpha = 2, N_\beta[4, 10[$ no θ