1 Momentum distributions

2 Second quantization

This section will be somewhat over-elaborated. But it can serve as a recapitulation of second quantization.

The one body momentum distribution operator is defined as,

$$\hat{n}(p) = \frac{1}{(2\pi)^3} \int \mathrm{d}^2\Omega_{\mathbf{p}} a^\dagger_{\mathbf{p}} a_{\mathbf{p}} \tag{1}$$

It's action on a multi particle ground state $|\Phi\rangle$,

$$\langle\Phi|\hat{n}(p)|\Phi\rangle = \frac{1}{(2\pi)^3} \int \mathrm{d}^2\Omega_{\mathbf{p}} \langle\Phi|a^\dagger_{\mathbf{p}} a_{\mathbf{p}}|\Phi\rangle \tag{2}$$

The creation and annihilation operators $a^\dagger_{\mathbf{p}}$,$a_{\mathbf{p}}$ have only meaning working on particles with definite momentum or the vacuum state $|0\rangle$.

$$\langle\Phi|a^\dagger_{\mathbf{p}} a_{\mathbf{p}}|\Phi\rangle = \int \mathrm{d}^3\mathbf{p}_1 \dots \mathrm{d}^3\mathbf{p}_A \langle\Phi|\mathbf{p}_1\mathbf{p}_2\dots\mathbf{p}_A\rangle \langle\mathbf{p}_1\mathbf{p}_2\dots\mathbf{p}_A|a^\dagger_{\mathbf{p}} a_{\mathbf{p}}|\Phi\rangle \tag{3}$$

$$= \int \mathrm{d}^A\mathbf{p}_1 \dots \mathrm{d}^3\mathbf{p}_A \langle\Phi|\mathbf{p}_1\mathbf{p}_2\dots\mathbf{p}_A\rangle \langle 0|a_{\mathbf{p}_1} a_{\mathbf{p}_2} \dots a_{\mathbf{p}_A} a^\dagger_{\mathbf{p}} a_{\mathbf{p}}|\Phi\rangle \tag{4}$$

Using the anticommutation relation $\{a_{\mathbf{p}}, a^\dagger_{\mathbf{q}}\} = \delta(\mathbf{p}-\mathbf{q})$, we get

$$\langle 0|a_{\mathbf{p}_1} a_{\mathbf{p}_2} \dots a_{\mathbf{p}_A} a^\dagger_{\mathbf{p}} a_{\mathbf{p}}|\Phi\rangle = \langle 0|a_{\mathbf{p}_1} a_{\mathbf{p}_2} \dots \delta(\mathbf{p}-\mathbf{p}_A) a_{\mathbf{p}}|\Phi\rangle - \langle 0|a_{\mathbf{p}_1} a_{\mathbf{p}_2} \dots a_{\mathbf{p}_{A-1}} a^\dagger_{\mathbf{p}} a_{\mathbf{p}_A} a_{\mathbf{p}}|\Phi\rangle \tag{5}$$

$$= \delta(\mathbf{p}-\mathbf{p}_A) \langle\mathbf{p}_1\mathbf{p}_2\dots\mathbf{p}|\Phi\rangle - \delta(\mathbf{p}-\mathbf{p}_{A-1}) \langle 0|a_{\mathbf{p}_1} \dots a_{\mathbf{p}_{A-2}} a_{\mathbf{p}_A} a_{\mathbf{p}}|\Phi\rangle \tag{6}$$

$$+ \langle 0|a_{\mathbf{p}_1} \dots a_{\mathbf{p}_{A-2}} a^\dagger_{\mathbf{p}} a_{\mathbf{p}_{A-1}} a_{\mathbf{p}_A} a_{\mathbf{p}}|\Phi\rangle \tag{7}$$

$$= \delta(\mathbf{p}-\mathbf{p}_A) \langle\mathbf{p}_1\mathbf{p}_2\dots\mathbf{p}_A|\Phi\rangle + \delta(\mathbf{p}-\mathbf{p}_{A-1}) \langle\mathbf{p}_1\dots\mathbf{p}_{A-2}\mathbf{p}_{A-1}\mathbf{p}_A|\Phi\rangle \tag{8}$$

$$+ \langle 0|a_{\mathbf{p}_1} \dots a_{\mathbf{p}_{A-2}} a^\dagger_{\mathbf{p}} a_{\mathbf{p}_{A-1}} a_{\mathbf{p}_A} a_{\mathbf{p}}|\Phi\rangle \quad = \dots \tag{9}$$

$$= \sum_{i=1}^{A} \delta(\mathbf{p}-\mathbf{p}_i) \langle\mathbf{p}_1\dots\mathbf{p}_A|\Phi\rangle + (-1)^A \underbrace{\langle 0|a^\dagger_{\mathbf{p}} a_{\mathbf{p}_1} \dots a_{\mathbf{p}_A} a_{\mathbf{p}}|\Phi\rangle}_{=0} \tag{10}$$

Hence,

$$\langle\Phi|a^\dagger_{\mathbf{p}} a_{\mathbf{p}}|\Phi\rangle = \int \mathrm{d}^3\mathbf{p}_1 \dots \mathrm{d}^3\mathbf{p}_A \langle\Phi|\mathbf{p}_1\mathbf{p}_2\dots\mathbf{p}_A\rangle \sum_{i=1}^{A} \delta(\mathbf{p}-\mathbf{p}_i) \langle\mathbf{p}_1\mathbf{p}_2\dots\mathbf{p}_A|\Phi\rangle \tag{11}$$

If $|\Phi\rangle$ is a slater determinant of orthonormal single particle wave functions $|\phi_{\alpha_i}\rangle$ we get,

$$\langle\Phi|a^\dagger_{\mathbf{p}} a_{\mathbf{p}}|\Phi\rangle = \sum_{i=1}^{A} |\langle\mathbf{p}|\phi_{\alpha_i}\rangle|^2 = \sum_{i=1}^{A} \phi^\dagger_{\alpha_i}(\mathbf{p}) \phi_{\alpha_i}(\mathbf{p}) \tag{12}$$

Note that we also could have derived this result by instead of inserting the unit $\prod_{i=1}^{A} \mathrm{d}^3\mathbf{p}_i |\mathbf{p}_i\rangle \langle\mathbf{p}_i|$ we expand $|\Phi\rangle$ in terms of single particle creation operators,

$$a^\dagger_{\mathbf{p}} a_{\mathbf{p}} |\Phi\rangle = a^\dagger_{\mathbf{p}} a_{\mathbf{p}} |\alpha_1\alpha_2\dots\alpha_A\rangle = a^\dagger_{\mathbf{p}} a_{\mathbf{p}} a^\dagger_{\alpha_1} a^\dagger_{\alpha_2} \dots a^\dagger_{\alpha_A} |0\rangle \tag{13}$$

The commutation relations between $a_{\mathbf{p}}$ and a_{α_i} are easily derived by expanding a_{α_i} in momentum creation operators,

$$a^\dagger_{\alpha_i} = \int \mathrm{d}^3\mathbf{k}\phi_{\alpha_i}(\mathbf{k})a^\dagger_k \tag{14}$$

$$\Rightarrow a_{\mathbf{p}}a^\dagger_{\alpha_i} = \int \mathrm{d}^3\mathbf{k}\phi_{\alpha_i}(\mathbf{k})a_{\mathbf{p}}a^\dagger_{\mathbf{k}} = \phi_{\alpha_i}(\mathbf{p}) - a^\dagger_{\alpha_i}a_{\mathbf{p}} \tag{15}$$

So,

$$a_{\mathbf{p}}\left|\Phi\right\rangle = a_{\mathbf{p}}a^\dagger_{\alpha_1}a^\dagger_{\alpha_2}\ldots a^\dagger_{\alpha_A}\left|0\right\rangle = (\phi_{\alpha_1}(\mathbf{p}) - a^\dagger_{\alpha_1}a_{\mathbf{p}})a^\dagger_{\alpha_2}\ldots a^\dagger_{\alpha_A}\left|0\right\rangle \tag{16}$$

$$= \sum_{i=1}^{A}(-1)^{i-1}\phi_{\alpha_i}(\mathbf{p})\left|\alpha_1\ldots\alpha_{i-1}\alpha_{i+1}\ldots\alpha_A\right\rangle \tag{17}$$

The conjugate gives,

$$\left\langle\Phi\right| a^\dagger_{\mathbf{p}} = \sum_{j=1}^{A}(-1)^{j-1}\left\langle\alpha_1\ldots\alpha_{j-1}\alpha_{j+1}\ldots\alpha_A\right|\phi^\dagger_{\alpha_j}(\mathbf{p}) \tag{18}$$

Hence,

$$\langle\Phi|a^\dagger_{\mathbf{p}}a_{\mathbf{p}}|\Phi\rangle = \sum_{i,j=1}^{A}(-1)^{i+j}\phi^\dagger_{\alpha_j}(\mathbf{p})\phi_{\alpha_i}(\mathbf{p})\underbrace{\langle\alpha_1\ldots\alpha_{j-1}\alpha_{j+1}\ldots\alpha_A|\alpha_1\ldots\alpha_{i-1}\alpha_{i+1}\ldots\alpha_A\rangle}_{=\delta_{ij}} \tag{19}$$

$$= \sum_i \phi^\dagger_{\alpha_i}(\mathbf{p})\phi_{\alpha_i}(\mathbf{p}) \tag{20}$$

Which is exactly the same result as before.

So the one body momentum distribution is given by,

$$\langle\Phi|\hat{n}(p)|\Phi\rangle = \sum_{i=1}^{A}\frac{1}{(2\pi)^3}\int \mathrm{d}^2\Omega_{\mathbf{p}}\phi^\dagger_{\alpha_i}(\mathbf{p})\phi_{\alpha_i}(\mathbf{p}) \tag{21}$$

Note that this distribution is normed to the number of particles A. To get the momentum distribution normed to unity we have to divide by A,

$$\langle\Phi|\hat{n}(p)|\Phi\rangle = \frac{1}{A}\sum_{i=1}^{A}\frac{1}{(2\pi)^3}\int \mathrm{d}^2\Omega_{\mathbf{p}}\phi^\dagger_{\alpha_i}(\mathbf{p})\phi_{\alpha_i}(\mathbf{p}) \tag{22}$$

3 Nucleus

3.1 shell.h

This class contains the quantum number of a shell nlj. It has two (proton & neutron) static arrays containing all the shells.

```
shellsN = [ Shell(n1,l1,j1), Shell(n2,l2,j2), ... ]
shellsP = [ Shell(n1,l1,j1), Shell(n2,l2,j2), ... ]
```

These two arrays are initialised and deleted by the static methods `Shell::initialiseShells`, `Shell::deleteShells`.

3.2 nucleus.h

First important method here is `Nucleus::makePairs`. Note that this relies on overloaded virtual functions to function. It iterates over the quantum numbers, $n_1 l_1 j_1 m_{j_1}, n_2 l_2 j_2 m_{j_2}$ and makes a pair for each of these combinations: `Pair::Pair(mosh,n1,l1,j1,mj1,t1,n2,l2,j2,mj2,t2)`. `mosh` is the return value of `RecMosh::createRecMosh(n1,l1,n2,l2,inputdir,outputdir)`, being a `RecMosh` object. The moshinsky brackets $\langle n_1 l_1 n_2 l_2; \Lambda | nlNL; \Lambda \rangle$ can be accessed by calling `RecMosh::getCoefficient(n,l,N,L,Lambda)`. Open shells are taken care of by calculating a open shell correction factor and applying it to the pair via `Pair::setfnorm(factor)`.

Once the pairs (`Pair::Pair`) are generated we can generate a

4 Pair coupling

4.1 pair.h

This class represents the state

$$|\alpha_1, \alpha_2\rangle_{\text{nas}} \quad , |\alpha\rangle \equiv |nljm_j tm_t\rangle \tag{23}$$

The class calculates all the coefficients,

$$C^A_{\alpha_1\alpha_2} = \langle A \equiv \{nlSjm_j, NLM_L TM_T\} | \alpha_1 \alpha_2 \rangle \tag{24}$$

The main method here is `Pair::makecoeflist()`. It loops over all possible values of $A \equiv \{S, T, n, l, N, M_L, j, m_j\}$. Where in the summation over $\{n, l, N, L\}$ the energy conservation $2n_1 + l_1 + 2n_2 + l_2 = 2n + l + 2N + L$ is taken into account to eliminate one of the summation loops, $L = 2n_1 + l_1 + 2n_2 + l_2 - 2n - l - 2N$. Note that M_T is also fixed by $M_T = m_{t_1} + m_{t_2}$ and no summation over this is performed, as we want to keep the contribution from different pairs seperated. For each A a new object `Newcoef` is generated and stored in the member `std::vector<NewCoef*> coeflist`.

4.2 newcoef.h

This class takes the parameters $n_1 l_1 j_1 m_{j_1} m_{t_1} n_2 l_2 j_2 m_{j_2} m_{t_2} NLM_L nlSjm_j TM_T$, and calculates the coefficient given in Eq. (24). It takes also a pointer to a `RecMosh` object that holds the Moshinsky brakets. The only function in this class is to calculate $C^A_{\alpha_1\alpha_2}$ using the formula,

$$\sum_{JM_J} \sum_{\Lambda} [1 - (-1)^{L+S+T}] \, \langle t_1 m_{t_1} t_2 m_{t_2} | TM_T \rangle \, \langle j_1 m_{j_1} j_2 m_{j_2} | JM_J \rangle \, \langle jm_j LM_L | JM_J \rangle$$
$$\langle nlNL; \Lambda | n_1 l_1 n_2 l_2; \Lambda \rangle_{\text{SMB}} \sqrt{2\Lambda + 1} \sqrt{2j+1} \begin{Bmatrix} j & L & J \\ \Lambda & S & l \end{Bmatrix}$$
$$\sqrt{2j_1+1}\sqrt{2j_2+1}\sqrt{2S+1}\sqrt{2\Lambda+1} \begin{Bmatrix} l_1 & s_1 & j_1 \\ l_2 & s_2 & j_2 \\ \Lambda & S & J \end{Bmatrix} \tag{25}$$

It is easy to check that the result indeed depends on α_1, α_2, A. Note that it is always assumed that $s_i, t_i \equiv \frac{1}{2}$ as we are dealing with protons and neutrons. This class also defines a `''key''` to be able to index the coefficients, `key = ''nlSjm_j.NLM_L.TM_T''`.

4.3 paircoef.h

This is a very thin class designed to do some bookkeeping. As outlined in Maartens thesis pg 156, different $|\alpha_1\alpha_2\rangle$ combinations will sometimes map to the same "rcm" states $A = |nlSjm_j NLM_L TM_T\rangle$. In matrix element calculations,

$$\langle \alpha_1 \alpha_2 | \hat{\mathcal{O}} | \alpha_1 \alpha_2 \rangle = \sum_{AB} C^{A\dagger}_{\alpha_1\alpha_2} C^B_{\alpha_1\alpha_2} \langle A | \hat{\mathcal{O}} | B \rangle \tag{26}$$

We want to calculate matrix elements as $\langle A|\hat{\mathcal{O}}|B\rangle$ only once. $|\alpha_1\alpha_2\rangle$ that map to the same A, B states should lookup the earlier calculated values for $\langle A|\hat{\mathcal{O}}|B\rangle$. In general the matrix element $\langle A|\hat{\mathcal{O}}|B\rangle$ is not diagonal. A `Paircoef` object has all the quantum numbers in a rcm state A. In addition it holds a value and a map `std::map<Paircoef*, double>`. The map is used to link a rcm state $|A\rangle$ to all other rcm states $|B\rangle$ which yield a non zero contribution for $\langle A|\hat{\mathcal{O}}|B\rangle$. The value for the transformation coefficients $C^{A,\dagger}_{\alpha_1,\alpha_2} C^{B}_{\alpha_1,\alpha_2}$ is stored in the second field of the map (`double`). So that the the summation over B (Eq. 26) is replaced by,

$$\langle\alpha_1\alpha_2|\hat{\mathcal{O}}|\alpha_1\alpha_2\rangle = \sum_{A} \sum_{\texttt{Paircoef(A).links}} \texttt{link.strength}\, \langle A|\hat{\mathcal{O}}|B\rangle \tag{27}$$

`Paircoef::add(double val)` adds `val` to private member `value` but as far as I can see this private member `value` is NEVER used!

5 Matrix Elements

First some theory on the matrix elements. In the calculation of the norm we only have the correlation operator $\hat{\imath}$ between the bras and kets.

$$\langle\alpha\beta|\hat{\imath}(\vec{x}_1,\vec{x}_2) + \hat{\imath}^\dagger(\vec{x}_1,\vec{x}_2) + \hat{\imath}^\dagger(\vec{x}_1,\vec{x}_2)\hat{\imath}(\vec{x}_1,\vec{x}_2)|\alpha\beta\rangle$$

$\hat{\imath}$ contains a central, tensor and spin-isospin part,

$$\hat{\imath}(\vec{x}_1,\vec{x}_2) = -f_c(r_{12}) + f_{t\tau}(r_{12})\hat{S}_{12}\hat{\vec{\tau}}_1\cdot\hat{\vec{\tau}}_2 + f_{\sigma\tau}(r_{12})\hat{\vec{\sigma}}_1\cdot\hat{\vec{\sigma}}_2\hat{\vec{\tau}}_1\cdot\hat{\vec{\tau}}_2\,.$$

Transforming to the c.m. and relative coordinates a general matrix-element term can be written as,

$$\langle n(lS)jm_j NLM_L TM_T|\hat{\mathcal{O}}^{p\dagger} f_p^\dagger f_q \hat{\mathcal{O}}^q|n'(l'S')j'm'_j N'L'M'_L T'M'_T\rangle$$

With $f_{p,q}\in\{1, f_c, f_{t\tau}, f_{\sigma\tau}\}$ and $\hat{\mathcal{O}}^{p,q}$ the corresponding operator $\in\{\mathbb{1},\mathbb{1},\hat{S}_{12}\hat{\vec{\tau}}_1\cdot\hat{\vec{\tau}}_2, \hat{\vec{\sigma}}_1\cdot\hat{\vec{\sigma}}_2\hat{\vec{\tau}}_1\cdot\hat{\vec{\tau}}_2\}$. As no operators act on the c.m. part $|NLM_L\rangle$ here we have,

$$\delta_{NN'}\delta_{LL'}\delta_{M_L M'_L}\,\langle n(lS)jm_j TM_T|\hat{\mathcal{O}}^{p\dagger} f_p^\dagger f_q \hat{\mathcal{O}}^q|n'(l'S')j'm'_j T'M'_T\rangle$$

6 Matrix elements bis

Let us take a look at

$$\langle S|\hat{\vec{\sigma}}_1\cdot\hat{\vec{\sigma}}_2|S'\rangle = 4\,\langle S|\hat{\vec{s}}_1\cdot\hat{\vec{s}}_2|S'\rangle = 4\,\langle S|\hat{\vec{S}}^2 - \hat{\vec{s}}_1^2 - \hat{\vec{s}}_2^2|S'\rangle = 2(S(S+1) - \frac{3}{4} - \frac{3}{4})\delta_{SS'} = \delta_{SS'}(2S(S+1)-3)$$

As we have 2 spin 1/2 particles S can be either $0, 1$ resulting in $\langle 1|\hat{\vec{\sigma}}_1\cdot\hat{\vec{\sigma}}_2|1\rangle = 1$, $\langle 0|\hat{\vec{\sigma}}_1\cdot\hat{\vec{\sigma}}_2|0\rangle = -3$.

Note that in the Maartens code the expression is modified to $4S-3$, which is equivalent for $S\in\{0,1\}$.

Exactly the same derivation can be made for $\hat{\vec{\tau}}_1\cdot\hat{\vec{\tau}}_2$ leading to the same result.

6.1 norm_ob : public operator_virtual_ob

Here we take a look at the calculation of the norm $\mathcal{N}$ in `norm_ob.cpp`. Note that this class inherits from `operator_virtual_ob`, declaring general one body member functions.

- `norm_ob::get_me( Pair )`. This calculates the matrix element **meanfield** matrix element sum

1. $\frac{2}{A}\sum_{AB} C^{A\dagger}_{\alpha_1\alpha_2} C^{B}_{\alpha_1\alpha_2} \langle A|B\rangle$ for a pp and/or nn pair(s) (isospin $M_T = \pm 1$)
2. $\frac{1}{A}\sum_{AB} C^{A\dagger}_{\alpha_1\alpha_2} C^{B}_{\alpha_1\alpha_2} \langle A|B\rangle$ for a pn pair (isospin $M_T = 0$)

for a specific pair $\alpha_1\alpha_2$ passed trough `Pair`.

For now I have no clue why/how the factor $\frac{2}{A}(\frac{1}{A})$ in front of $\sum_{AB} C^{A\dagger}_{\alpha_1\alpha_2} C^{B}_{\alpha_1\alpha_2} \langle A|B\rangle$ ...

It is possible to filter on relative quantum numbers on n_A, l_A, n_B, l_B, selecting specific contributions `nAs,lAs,nBs,lBs` to the sum. A value of -1 for these variables is interpreted as "all values allowed". Trough the braket $\langle A|B\rangle$ we already have $n_A = n_B := n$, $l_A = l_B := l$.

- `if (nAs > -1 && nBs > -1)` This forces `nAs` $=$ `nBs` $= n$. So for `nAs` $\neq$ `nBs` we will get 0.
- `if (nAs == -1 && nBs > -1)` This forces `nBs` $= n$. Selecting a specific $n = n_A = n_B$ contribution.
- `if (nAs > -1 && nBs == -1)` This forces `nAs` $= n$. Selecting a specific $n = n_A = n_B$ contribution.
- `if (nAs == -1 && nBs == -1)` This makes no restrictions on $n = n_A = n_B$.

The exact same is valid for $l = l_A = l_B$ and `lAs,lBs`. A few examples (`nAs,lAs,nBs,lBs`):

- (`-1, 2,-1,-1`) : allow all $n = n_A = n_B$ values. Restriction on $l = l_A = l_B =$ `2`.
- (`-1, 2,-1, 2`) : allow all $n = n_A = n_B$ values. Restriction on $l = l_A = l_B =$ `2`.

As the unrestricted sum $\sum_{AB} C^{A\dagger}_{\alpha_1\alpha_2} C^{B}_{\alpha_1\alpha_2} \langle A|B\rangle = \sum_A |C^{A}_{\alpha_1\alpha_2}|^2$ equals 1, the return value of `get_me` (for the unrestricted sum) is,

- $\frac{2}{A}$ with no restriction on the isospin (`norm_ob::norm_ob_params.t = 0`)
- $\frac{2}{A}$ for pp-pairs, $\frac{1}{A}$ for pn-pairs and 0 for nn-pairs for a proton (`norm_ob::norm_ob_params.t = 1`)
- 0 for pp-pairs, $\frac{1}{A}$ for pn-pairs and $\frac{2}{A}$ for nn-pairs for a neutron (`norm_ob::norm_ob_params.t =-1`)

If we sum over all the pairs $\sum_{\texttt{pair in pairs}}$ `norm::ob_get_me(pair,...)` we get,

- $\frac{A(A-1)}{2}\frac{2}{A} = A-1$ with no restriction on the isospin (`norm_ob::norm_ob_params.t = 0`)
- $\frac{Z(Z-1)}{2}\frac{2}{A} + NZ\frac{1}{A} + \frac{N(N-1)}{2}0 = Z\frac{A-1}{A}$ for a proton (`norm_ob::norm_ob_params.t = 1`)
- $\frac{Z(Z-1)}{2}0 + NZ\frac{1}{A} + \frac{N(N-1)}{2}\frac{2}{A} = N\frac{A-1}{A}$ for a neutron (`norm_ob::norm_ob_params.t =-1`)

Open shellness not taken into account here. Must be done somewhere else (higher up)...
For closed shell nuclei everything seems fine. For open shells however we get some strange results. For example ^{27}Al with 13 protons and 14 neutrons has an open $1d\frac{5}{2}$ proton shell. Open-shell nuclei are treated as closed shell but the pairs in the open shells get a weight factor. This weight factor however is **not** present in the method `norm::ob_get_me(pair,...)`. Hence as $A = 27$ but the closed shell equivalent with $A = 28$ causes the number of pairs to be $28 \cdot 27/2$ instead of $27 \cdot 26/2$. We get

- $\frac{28\cdot 27}{2}\frac{2}{27} = 28$ (`norm_ob::norm_ob_params.t = 0`)
- $\frac{14\cdot 13}{2}\frac{2}{27} + \frac{14\cdot 14}{27} = \frac{378}{27} = 14$ (`norm_ob::norm_ob_params.t = 1`)
- $\frac{14\cdot 14}{27} + \frac{14\cdot 13}{2}\frac{2}{27} = \frac{378}{27} = 14$ (`norm_ob::norm_ob_params.t =- 1`)

- `norm_ob::get_me_corr_right( Pair )`.