1 Pair coupling

1.1 pair.h

This class represents the state

$$|\alpha_1, \alpha_2\rangle_{\mathrm{nas}} \quad , |\alpha\rangle \equiv |nljm_jtm_t\rangle \tag{1}$$

The class calculates all the coefficients,

$$C^A_{\alpha_1\alpha_2} = \langle A \equiv \{nlSjm_j, NLM_LTM_T\}|\alpha_1\alpha_2\rangle \tag{2}$$

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`.

1.2 newcoef.h

This class takes the parameters $n_1l_1j_1m_{j_1}m_{t_1}n_2l_2j_2m_{j_2}m_{t_2}NLM_LnlSjm_jTM_T$, and calculates the coefficient given in Eq. (2). 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,

$$\begin{aligned}\sum_{JM_J}\sum_{\Lambda}[1-(-1)^{L+S+T}] \langle t_1m_{t_1}t_2m_{t_2}|TM_T\rangle \langle j_1m_{j_1}j_2m_{j_2}|JM_J\rangle \langle jm_jLM_L|JM_J\rangle \\ \langle nlNL;\Lambda|n_1l_1n_2l_2;\Lambda\rangle_{\mathrm{SMB}} \sqrt{2\Lambda+1}\sqrt{2j+1}\left\{\begin{array}{ccc} j & L & J \\ \Lambda & S & l \end{array}\right\} \\ \sqrt{2j_1+1}\sqrt{2j_2+1}\sqrt{2S+1}\sqrt{2\Lambda+1}\left\{\begin{array}{ccc} l_1 & s_1 & j_1 \\ l_2 & s_2 & j_2 \\ \Lambda & S & J \end{array}\right\}\end{aligned} \tag{3}$$

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’’`.

2 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 $A = |nlSjm_jNLM_LTM_T\rangle$. In matrix element calculations,

$$\langle\alpha_1\alpha_2|\hat{\mathcal{O}}|\alpha_1\alpha_2\rangle = \sum_{AB} C^{A\dagger}_{\alpha_1\alpha_2} C^B_{\alpha_1\alpha_2} \langle A|\hat{\mathcal{O}}|B\rangle \tag{4}$$

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. 4) is replaced by,

$$\langle\alpha_1\alpha_2|\hat{\mathcal{O}}|\alpha_1\alpha_2\rangle = \sum_A \sum_{\texttt{Paircoef(A).links}} \texttt{link.strength} \langle A|\hat{\mathcal{O}}|B\rangle \tag{5}$$

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