Computer Code for Calculating Matrix Elements and Overlaps of States for the Generalized Seniority Scheme via Recurrence Relations

Ke Cai

Bard College

2010 NSF/REU Program

Physics Department, University of Notre Dame

Advisor: Mark Caprio

August 6th, 2010

Abstract

The generalized seniority approximation provides a truncation scheme for the nuclear shell model based on building the states of the nucleus from nucleon pairs. We present a computer code to calculate matrix elements of one-body and two-body operators between generalized seniority states and overlaps of these states based on a set of recurrence relations.

1 Introduction

The generalized seniority approximation provides a truncation scheme for the nuclear shell model. The nuclear interaction tends to induce nucleons to form pairs coupled to angular momentum zero. This approximation considers the ground state of an even-even nucleus to be consisted of collective S pairs, which are constructed from like nucleons coupled to angular momentum zero. Then the low-lying excited states of a nucleus are obtained from breaking one or more of these pairs. The generalized seniority, v, of the state of a nucleus, is defined to be the number of nucleons not participating the collective S pairs. To perform calculations based on the generalized seniority scheme, it is first necessary to compute the matrix elements of one-body and two-body operators between basis states of low generalized seniority. It is also necessary to calculate the overlaps between these states in order to construct an orthogonal basis. In this paper, we present a computer code to calculate such quantities. The main algorithm is based on a set of recurrence relations recently derived [1]. In Sec.2, we go over some essential definitions. In Sec.3, we briefly introduce the recurrence relations[1]. We present the structure of the code in Sec.4 and 5. In Sec.6, we present validations we have performed. In Sec.7, we summarize the paper and propose future applications of the code.

[1]For detailed discussion involving the contents of Sec.2 and Sec.3, please refer to [1]

2 Definitions

We first review some of the necessary definitions of generalized seniority scheme and notations used in the recurrence relations. Working in a single major shell, one can uniquely specify a single-particle level. Let $C^\dagger_{c,\gamma}$ be the creation operator [2] for a particle in the state of angular momentum c and z-projection quantum number γ. The angular-momentum coupled product of two spherical tensor operators is defined by $(A^a \times B^b)^c_\gamma = \sum_{\alpha\beta}(a\alpha b\beta|c\gamma)A^a_\alpha B^b_\beta$[3]. Then the angular-momentum coupled pair creation operator is defined by

$$A^{e\,\dagger}_{ab} = (C^\dagger_a \times C^\dagger_b)^e, \tag{1}$$

and the collective S pair is defined as the following linear combination of pairs of nucleons coupled to angular momentum zero:

$$S^\dagger = \sum_c \alpha_c \frac{\hat{c}}{2} A^{0\,\dagger}_{cc}, \tag{2}$$

with $\hat{c} = (2c+1)^{1/2}$. The amplitudes α_c are conventionally taken subject to the normalization condition $\sum_c (2c+1)\alpha_c^2 = \sum_c (2c+1)$. The state of zero generalized seniority and consisted of N S-pairs is defined by the S-pair successively acting on the vacuum N times: $|\,S^N\rangle = S^{\dagger\,N}|\,0\rangle$. In general, a state of generalized seniority v is constructed as as $|\,S^N F^f\rangle = S^{\dagger\,N} F^{f\,\dagger}|\,0\rangle$, where $F^{f\,\dagger}$ is a cluster of v fermion creation operators coupled to angular momentum f.

To calculate matrix elements of one-body and two-body operators, quantities of immediate interest are reduced matrix elements $\langle S^N G^g \,\|\, T^t_{rs} \,\|\, S^N F^f\rangle$, taken between generalized seniority states, where T^t_{rs} is the elementary one-body multipole operators, and F^f and G^g represent clusters of v nucleons not participating in the S pairs. Since generalized seniority states are not orthonormal, we also need to calculate the overlaps between states :

[2] Here we do not distinguish between the level c and its angular momentum j_c. However, when we start to work with the code, we labeled single particle levels using integers and store corresponding angular momenta elsewhere, for simplicity of storing information. For definition and further discussion of creation and annihilation operators, see [2]

$\langle S^N G^f \mid S^N F^f \rangle$. In [1], $\Gamma_N^{(v)}[\cdots]$ is used for the one-body operator matrix elements between states of equal generalized seniority, *e.g.*,

$$\Gamma_N^{(4)}[(cd)^f(kl)^{n\,h}|(rs)^t|(ab)^e(ij)^{m\,g}] \equiv \langle S^N(A^f_{cd}A^n_{kl})^h \parallel T^t_{rs} \parallel S^N(A^e_{ab}A^m_{ij})^g\rangle, \tag{3}$$

and $\Phi_N^{(v)}[\cdots]$ for the overlaps, *e.g.*,

$$\Phi_N^{(4)}[(cd)^f(kl)^{n\,g}|(ab)^e(ij)^{m\,g}] \equiv \langle S^N(A^f_{cd}A^n_{kl})^g \mid S^N(A^e_{ab}A^m_{ij})^g\rangle. \tag{4}$$

We will follow the same notations in this paper.

Since a state with any generalized seniority and N number of S pairs (N>0) can be expressed as a linear combination of states with higher generalized seniority by expanding one or more S pairs in terms of A^0_{kk}, matrix elements or overlaps involving states of different generalized seniority can be calculated using states with the same generalize seniority, *e.g.*,

$$\langle S^N A^g_{cd}|S^{N-1}(A^e_{ab}A^m_{ij})^g\rangle = \sum_k \alpha_k \frac{\hat{k}}{2}\langle S^{N-1}(A^0_{kk}A^g_{cd})^g|S^{N-1}(A^e_{ab}A^m_{ij})^g\rangle. \tag{5}$$

It is noteworthy that $\Gamma_N^{(v)}$ and $\Phi_N^{(v)}$ obey some symmetry relations under permutation of arguments, *e.g.*,

$$\begin{aligned}\Gamma_N^{(4)}[(cd)^f(kl)^{n\,h}|(rs)^t|(ab)^e(ij)^{m\,g}] &= -\theta(abe)\Gamma_N^{(4)}[(cd)^f(kl)^{n\,h}|(rs)^t|\underbrace{(ba)}^{e}(ij)^{m\,g}] \\ &= \theta(emg)\Gamma_N^{(4)}[(cd)^f(kl)^{n\,h}|(rs)^t|\underbrace{(ij)^{m}(ab)}^{e\,g}].\end{aligned} \tag{6}$$

or complex conjugation, *e.g.*,

$$\Gamma_N^{(4)}[(cd)^f(kl)^{n\,h}|(rs)^t|(ab)^e(ij)^{m\,g}] = -\theta(rsgh)\Gamma_N^{(4)}[\underbrace{(ab)^e(ij)^{m\,g}}|(sr)^t|(cd)^f(kl)^{n\,h}]. \tag{7}$$

3 The Recurrence Relations

In [1], the reduced matrix elements of the one-body operator and overlaps between generalized seniority states are written as vacuum expectation values, *e.g.*,

$$\langle S^N G^g \parallel T^t_{rs} \parallel S^N F^f \rangle = (-)^{f-t-g} \langle 0|(\tilde{G}^g \tilde{S}^N \times T^t_{rs} \times S^{\dagger\, N} F^{f\,\dagger})^0|0\rangle, \tag{8}$$

and

$$\langle S^N G^f \mid S^N F^f \rangle = \hat{f}^{-1} \langle 0|(\tilde{G}^f \tilde{S}^N \times S^{\dagger\, N} F^{f\,\dagger})^0|0\rangle. \tag{9}$$

The recurrence relations for the matrix elements are derived by first commuting the one-body operator T to the right and then a pair creation operator A to the left to annihilate the vacuums (shown by the arrows), resulting in terms due to commutators (shown beneath the arrows), which now involve states with lower generalized seniority or lower N, schematically:

$$\begin{aligned}
\Gamma^{(v)}_N &\sim \langle 0|(\tilde{G}\tilde{S}^N)\, \overrightarrow{T\,(S^{\dagger\, N} F^\dagger)}|0\rangle \\
&\qquad\qquad\qquad\qquad\quad {\scriptstyle A^\dagger S^{\dagger\, N-1}} \\
&\sim \langle 0|(\tilde{G}\tilde{S}^N)\,(S^{\dagger\, N} \overrightarrow{T F^\dagger})|0\rangle + \langle 0|\tilde{G}\tilde{S}^N A^\dagger S^{\dagger\, N-1} F^\dagger|0\rangle \\
&\qquad\qquad\qquad\quad {\scriptstyle H^\dagger \sim [T, F^\dagger]} \\
&\sim \langle 0|(\tilde{G}\tilde{S}^N)\,(S^{\dagger\, N} H^\dagger)|0\rangle + \langle 0|\tilde{G}\tilde{S}^N A^\dagger S^{\dagger\, N-1} F^\dagger|0\rangle.
\end{aligned} \tag{10}$$

and then

$$\begin{aligned}
&\langle 0|\tilde{G}\tilde{S}^N A^\dagger S^{\dagger\, N-1} F^\dagger|0\rangle \\
&\quad {\scriptstyle \tilde{S}^{N-1} + T\tilde{S}^{N-1} + \tilde{S}^{N-1} T} \\
&\qquad \sim \langle 0|(\overleftarrow{\tilde{G}A^\dagger}\tilde{S}^N)\,(S^{\dagger\, N-1} F^\dagger)|0\rangle + \langle 0|(\tilde{G}\tilde{S}^{N-1})\,(S^{\dagger\, N-1} F^\dagger)|0\rangle \\
&\qquad\qquad\quad {\scriptstyle \tilde{I} \sim [\tilde{G}, A^\dagger]} \\
&\qquad\qquad + \langle 0|(\overleftarrow{\tilde{G}T}\tilde{S}^{N-1})\,(S^{\dagger\, N-1} F^\dagger)|0\rangle + \langle 0|(\tilde{G}\tilde{S}^{N-1})\,T\,(S^{\dagger\, N-1} F^\dagger)|0\rangle \\
&\qquad\qquad\quad {\scriptstyle \tilde{E} \sim [\tilde{G}, T]} \\
&\qquad \sim \langle 0|(\tilde{I}\tilde{S}^N)\,(S^{\dagger\, N-1} F^\dagger)|0\rangle + \langle 0|(\tilde{G}\tilde{S}^{N-1})\,(S^{\dagger\, N-1} F^\dagger)|0\rangle \\
&\qquad\qquad + \langle 0|(\tilde{E}\tilde{S}^{N-1})\,(S^{\dagger\, N-1} F^\dagger)|0\rangle + \langle 0|(\tilde{G}\tilde{S}^{N-1})\,T\,(S^{\dagger\, N-1} F^\dagger)|0\rangle.
\end{aligned} \tag{11}$$

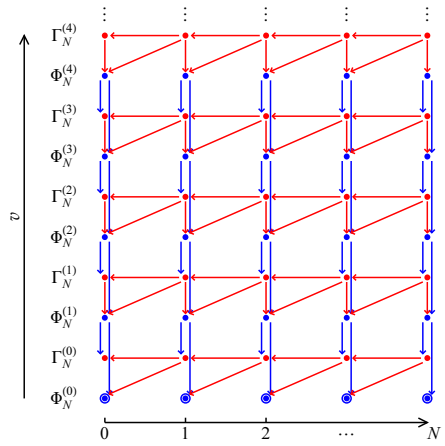
The recurrence relations for $\Phi_N^{(v)}$ are derived along the same line. The resulting recurrence relations have the following forms:

$$\Gamma_N^{(v)} \sim \Phi_N^{(v)} + \Gamma_{N-1}^{(v)} + \Phi_{N-1}^{(v)}. \tag{12}$$

and

$$\Phi_N^{(v)} \sim \Gamma_N^{(v-1)} + \Phi_N^{(v-1)}. \tag{13}$$

The recurrence network of $\Gamma_N^{(v)}$ and $\Phi_N^{(v)}$ are shown below.



The base case of the recurrence network – the overlaps $\Phi_k^{(0)}$ $(k = 0, 1, \ldots, N)$, have been calculated, using combinatorial methods[4][5]:

$$\Phi_N^{(0)} = (N!)^2 \sum_{(M_c)\in\mathcal{P}(N,D)} \left[\prod_c \alpha_c^{2M_c} \binom{\Omega_c}{M_c} \right]. \tag{14}$$

4 Structure of the Code

To preserve the algebraic structure of our problem, we construct various classes to represent objects we work with. The following classes are key to the structure of the code.

4.1 The `halfint` Class

As shown by Eq.(3) and (4), all the calculations of matrix elements and overlaps between states are carried out using angular momenta of individual or coupled nucleons, which can only have integer or half-integer values (e.g., -1, -$\frac{1}{2}$, 0, $\frac{1}{2}$, 1, etc.,). Since the C++ standard library does not define a data type for fractions, we first construct the class `halfint` , which includes both integers and half-integers (an integer is considered a half-integers with an even numerator). `halfint` only defines one member, the `numerator`, which is of type `int`, in C++. Without defining a denominator, we assume that the denominator of any object is 2. We then define appropriate operations of `halfint` type, such as addition, subtraction, comparison, etc., in terms of `numerator` of `halfint`, i.e. twice the actual value of `halfint` objects. Now a user of the code can easily preserve half-integer properties of angular momenta during computation and perform calculations involving them intuitively. In addition to the operations above, a series of other functions associated with angular momenta are defined. Variables of `halfint` type can now be used as building blocks for the classes to follow.

4.2 The `cluster` Class

We now construct the `cluster` class to represent clusters, in consistency with their physical definition in Sec.2. The `cluster` class has four members, for storing four kinds of information of the nucleons being created or annihilated: their species (neutrons or protons), the single particle levels on which they are created or annihilated, intermediate angular momenta resulting from coupling pairs of nucleons and total angular momentum of all particles created or annihilated. In particular, intermediate and total angular momenta are declared to be of type `halfint`.

In order to use the symmetric property mentioned in Sec.2, we define the canonical ordering of a cluster and a function to arrange a cluster into such an order and multiplying the cluster with a phase factor. The canonical order of a cluster is defined such that the single particle indices are in ascending order within pairs, pairs of indices of single particle levels are in ascending order lexicographically, i.e. $(ab) < (cd)$ (where (ab) and (cd) are ordered pairs)

if $a < c$ or if $a = c$ and $b < d$. If two pairs of single particle indices are equal, these two pairs are then arranged such that their coupled angular momenta are in ascending order. Then using the symmetries in Eq(6), clusters with arguments permuted with respect to each others can be related by a simple phase factor.

Other important functions defined for `cluster` include those that check the physical validity of grouped clusters, such as one that checks triangularity of coupled angular momenta. When an object of `cluster` is found to be "invalid", these functions either make the term involving such an invalid cluster zero, or terminate the program, which cut down the amount of calculation or avoid calculating physically impossible quantities. Using `cluster`, we construct the two following classes.

4.3 The `tensorOp` Class

`tensorOp` is constructed for spherical tensor operators. `tensorOp` has members `creation`, `annihilation` (of type `cluster`, defined for the creation-operator-cluster and annihilation-operator-cluster) and `opTotalAm` (of type `halfint`, defined for the coupled angular momentum of total angular momenta `totalAm` of `creation` and `annihilation`. One important operation defined for a `tensorOp` is taking it's adjoint, realized by swapping `creation` and `annihilation` and multiplying with an appropriate phase.

4.4 The `state` Class

Different from `tensorOp`, which uses `cluster` type as members, the class `state`, constructed for representing a generalized seniority state, is a derived class of `cluster`, with an additional member, `pairnumber`, which indicates the number of collective S pairs of the state. Since a generalized seniority state is defined as N S pair creation operators and a cluster of v creation operators acting on $|\,0\rangle$. our construction of a generalized seniority state is consistent with its physical definition in the way that $S^{\dagger\,N}$ is simplified as N, indicated by `pairnumber` and the notion of $|\,0\rangle$ is suppressed. As a derived class of `cluster`, `state` inherits a number of functions defined for `cluster`, while some other functions need to be

overloaded or redefined to take `pairnumber` into account. One function specially defined for `state` is `raiseSeniority` based on Eq(5), which returns a series of states with with higher seniority, lower `pairnumber` and appropriate coefficients. With this function, the code is able to calculate matrix elements and overlaps between states of different generalized seniority. It also provides means to internal consistency checks.

4.5 The `phiEnsemble` and the `gammaEnsemble` Classes

In recurrence calculations, some quantities of lower seniority or lower N come up repeatedly. To take advantage of this, we use the C++ container, `map`, to cache the those values. Each entry in a `map` is consisted of a unique `key` , and a corresponding value to the `key`. In our case, the value is any intermediate calculation of a matrix element or the overlap between two states. In order to uniquely identify quantities with `key`, we construct the classes `phiEnsemble` and `gammaEnsemble`, used to identify values of Φ's and Γ's, respectively. A `phiEnsemble` contains two members of type `state`, namely, `bra` and `ket`. This way, an object of `phiEnsemble` uniquely labels the overlap between its `bra` and `ket`. `gammaEnsemble` is defined and used in the same way, except it contains an additional member of type `tensorOp`, to indicate the spherical tensor operator involved in the calculation. To utilize symmetries due to complex conjugation, and provide a rule for C++ to look up values, we define the canonical form of an ensemble and a function to set an ensemble to "canonical form". The canonical form of an ensemble is a form such that both of the states involved are in "canonical order", and `bra` is "less" than `ket` (defined by lexicographically comparing each member of the two states). In the case of a `gammaEnsemble`, if `bra` and `ket` were the same, and `creation` were "greater than" `annihilation` in the spherical tensor operator (again, "greater than" is defined by lexicographical comparison), the adjoint of the spherical tensor operator would be taken, resulting in an appropriate phase.

The above five classes are the foundations of our code. With these classes, we are now able to perform specific calculations of matrix elements and overlaps in a straightforward

way.

5 Calculations and Caching

We used the recurrence relations as our main algorithm for calculating the matrix elements and overlaps. We developed two sets of functions. One set consists of specific functions, `Phi0`, `Phi1`, `Gamma0`, `Gamma1`, etc., defined for each seniority, calculating matrix elements and overlaps of states using the corresponding formulae provided by the recurrence relations. The other set consists of two general functions, `Gamma` and `Phi`, which perform preliminary works, such as checking the validity of states and operators, arranging ensembles into "canonical form", looking up existing values in the `map`'s, etc. After all the preliminary works are done, if , it calls the specific functions according to seniority of the states involved and finally caches the values in the map.

In side the specific set of functions, recurrence relations are realized by constructing new states of lower N or lower seniority, using the information in the original arguments, and calling the general functions with the newly constructed states or operators as arguments. These two sets of functions are related by calling each other in the recurrence network until both v and N reduce to 0.

6 Validations

Some of the formulae provided by the recurrence relation have many arguments and are error-prone. To check both the correctness of the code and of the formulae themselves, we are able to perform some validations.

6.1 Comparing with Combinatorial Methods

Overlaps between states of generalized seniority 2 were evaluated as sums over overlaps between states of generalized seniority 0 using combinatorial methods [1]. We compared our

results calculated by the recurrence formulae, to those given by the explicit formulae. The results agreed – this proves that both the derivation of the recurrence relations and the code are correct up to $v = 2$.

6.2 Internal Consistency Checks

As mentioned in Eq.(5), a state with N number of S pairs and generalized seniority v can be expanded as a linear combination of states with (N-1) number of S pairs generalized seniority $v - 2$. Based on this, we compared results calculated by upgrading both of the states involved in a specific calculation, *e.g.,*

$$\langle S^N A^g_{cd} | S^N A^e_{ab} \rangle = \sum_i \sum_j \alpha_i \alpha_j \frac{\hat{i}}{2} \frac{\hat{j}}{2} \langle S^{N-1} (A^0_{ii} A^g_{cd})^g | S^{N-1} (A^0_{jj} A^e_{ab})^e \rangle. \tag{15}$$

7 Summary

Based on the recurrence relations, the computer code we present here is able to calculate matrix elements and overlaps. We can use the results of the code to calculate matrix elements of tensor operators of physical interest, e.g. Hamiltonian and the electromagnetic transition operators. Planned applications of the code include testing pair structures and extending shell model calculations to nuclei which cannot easily be reached by conventional calculations, and studying the mapping of shell model onto the Interactive Boson Model[Reference!], which treats pairs of nucleons as composite bosons.

8 Acknowledgements

This work was supported by the US DOE under grant DE-FG02-95ER-40934.

References

[1] F.Q. Luo and M.A. Caprio, Nucl. Phys. A (2010).

[2] J. Suhonen, *From Nucleons to Nucleus* (Springer-Verlag, Berlin, 2007).

[3] D. A. Varshalovich, A. N. Moskalev, and V. K. Khersonskii, *Quantum Theory of Angular Momen- tum* (World Scientific, Singapore, 1988).

[4] Y. K. Gambhir, A. Rimini, and T. Weber, Phys. Rev. 188, 1573 (1969).

[5] S. Pittel, P. D. Duval, and B. R. Barrett, Ann. Phys. (N.Y.) 144, 168 (1982).