RCSFexpand: A new program for generating wave function expansions

Charlotte Froese Fischer

Computer Science Department, University of British Columbia

E-mail: cff@cs.ubc.ca

Abstract. An efficient new algorithm for generating wave function expansions for complex atoms is to be developed. The implementation in FORTRAN95 uses a data structure that defines the expansion as a list of configurations (CFGs) and their couplings. All sorting or merging is done at the CFG level. A flow diagram is provided.

1. Introduction to the problem

Currently, wave function expansions are obtained using RCSFGENERATE. This is a program for calculations where the order of the CSFs are not important. But for large expansions, it is important to be able to think of contributions as being large, such as those that define the MR set, and those that are small corrections with the large contributors preceding the small. For Lanthanides also, it was important to include only those small contributions that interact with at least one or more members of the MR set. This could be done by running scripts, and invoking a number of tools. The program RCSFexpansion is a program that does all this for the user.

The script below shows the problem that needs to be solved.

```
----------------------------------------
# Script for including CC
Z='Ce+2'; jb='0'; je='12'
rcsfgenerate <<EOF1
*
4
4d(10,*)4f(2,*)5s(2,*)5p(6,*)

 5s, 5p, 5d, 5f, 5g
 ${jb}, ${je}
0
n
EOF1
mv rcsf.out rcsfmr.inp
rcsfgenerate <<EOF1
*
4
4d(10,*)4f(2,*)5s(2,*)5p(6,*)
4d(10,*)4f(2,*)5s(2,*)5p(4,*)5d(2,*)
4d(10,*)5s(2,*)5p(6,*)5d(2,*)

 5s, 5p, 5d, 5f, 5g
 ${jb}, ${je}
0
n
EOF1
mv rcsf.out rcsf.inp
rcsfinteract <<EOF
1
EOF
mv rcsf.out rcsfmr.inp
rcsfgenerate <<EOF1
*
4
4d(10,*)4f(2,*)5s(2,*)5p(4,*)5d(2,*)
4d(10,*)4f(2,*)5s(2,*)5p(6,*)
4d(10,*)5s(2,*)5p(6,*)5d(2,*)
```

```
 5s, 5p, 5d, 5f, 5g
 ${jb}, ${je}
2
n
EOF1
mv rcsf.out rcsf.inp
rcsfinteract <<EOF
1
EOF
----------------------------------------
```

The script first determines the MR set that represents the most important CSFs of the expansion and to which all other CSFs will be required to interact. The last call to RCSFGENERATE has three CSFs from which SD excitations need to be generated, all to the same orbital set. We need to know how RCSFGENERATe deals with the merging of lists from the different CSFs. It has been found that the process is faster if the most complicated CSF is listed first. The number of CSFs generated from only the first CSF is more than 93 MILLION. For $J = 0$ the expansions are always shorter and it was found that, out of nearly 3 MILLION CSFs, only 1 in 33 interacted with the MR set. Clearly RCSFGENERATE should consider only CSFs that interact, in fact, subroutines of RCSFINTERACT should be part of RCSFGENERATE. In applying SD successively to three CSFs, the virtual orbital set is the same for all and the lists generated are merged.

The second script is an example of when the virtual orbital set increases in size. In this case, the second set of lists should be ADDED to the existing list and only CSFS that include extra orbitals should be included.

```
-----------------------------------
#Script with different orbital sets
Z='Ce+2'; jb='0'; je='12'

rcsfgenerate <<EOF1
*
4
4d(10,*)4f(2,*)5s(2,*)5p(6,*)
4d(10,*)4f(1,*)5s(2,*)5p(6,*)5f(1,*)
4d(10,*)5s(2,*)5p(6,*)5d(2,*)

 5s, 5p, 5d, 5f, 5g
 ${jb}, ${je}
2
y
4d(10,*)5s(2,*)5p(6,*)5d(2,*)

 6s, 6p, 6d, 6f, 6g, 6h
 ${jb}, ${je}
2
n
EOF1
```

```
cp rcsf.out ${Z}.5b.c
----------------------------------
```

Excitations in the first group of CSFs are all to the same orbital set and consequently the CSFs generated should be merged by configuration. The second set is different. In this case, only CSFs that included one or more of the added orbitals (6l, in this case) should be included and ADDED to the previous set.

In both of these examples, the lists should be viewed as lists of configurations (CFGs). All sorting and merging should be done at the CFG level with possibly information about coupling being included at the subsequent level.

In all cases, only CSFs that interact with the MR, should be included.

2. A Data Structure for a CFG

In the most readable and efficient form, the list of CSFs should be presented as a list of CFGs and their coupling. An example is the following for $4f^{10}4f^{4}5s^{2}5p^{2}5d^{2}$ coupled to $J = 0$. Note that $J = 0$ coupled states are typically an order of magnitude shorter that the high J quantum numbers.

```
-------------------------------------------------------------------
  4d-( 4)  4d ( 6)  4f ( 4)  5s ( 2)  5p ( 2)  5d-( 1)  5d ( 1)
                         0                 2      3/2      5/2
                                           2      5/2      0+
                    2;   2                 0      3/2      5/2
                                                  5/2      0+
                    2;   2                 2      3/2      5/2
                                           1      5/2      0+
                                           2      5/2      0+
                                           3      5/2      0+
                                           4      5/2      0+
                    2;   4                 0      3/2      5/2
                                                  5/2      0+
                    2;   4                 2      3/2      5/2
                                           2      5/2      0+
                                           3      5/2      0+
                                           4      5/2      0+
                         6                 2      3/2      5/2
                                           4      5/2      0+
                    4;   2                 0      3/2      5/2
                                                  5/2      0+
                    4;   2                 2      3/2      5/2
                                           1      5/2      0+
                                           2      5/2      0+
                                           3      5/2      0+
                                           4      5/2      0+
                    4;   4                 0      3/2      5/2
                                                  5/2      0+
                    4;   4                 2      3/2      5/2
                                           2      5/2      0+
```

```
                                    3       5/2       0+
                                    4       5/2       0+
                  5                 2      3/2       5/2
                                    3       5/2       0+
                                    4       5/2       0+
-----------------------------------------------------------------
```

In current RCSFGENERATE format the list repeats the configuration information for each CSF and also the subshell quantum numbers. Since Fortran2005, parameterized user defined derived data types can be declared and CFGs generated as a linked list where each node is a CFG with its parameters.

Associated with each CSF are three parameters:

symbol	definition	example
`n_sub`	the number of subshells	7
`n_qn`	the number of different subshell quantum numbers	11
`n_csf`	the number of CSFs	22

There also are a number of arrays:

symbol	definition
`iel(n_sub)`	indices to the electrons in the subshells
`qel(n_sub)`	occupation number for each subshell
`qns(n_qn, n_sub)`	For each set element,the quantum numbers for each subshell
`jcup(n_csf, n_sub`	For each csf, the coupling of its subshells
`set(n_qn)`	For each set of quantum numbers, the number of CSFs

The information about the quantum numbers of a subshell is encoded to include both the J quantum number and the seniority. The coupling of the final CSFs is in the order of sets, `i=1,n_qn`, where `set(i)` is the number of CSFs are associated with the `qns(i,1:m_sub)`.

A possible derived data type for CFG is the following:

```
TYPE  CFG(n\_sub, n\_qn, n\_csf)
     INTEGER, LEN ::  n\_sub, n\_qn, n\_csf
     INTEGER(kind=1), ALLOCATABLE. DIMENSION(n\_sub)        :: iel, qa
     INTEGER(kind=1), ALLOCATABLE, DIMENSION(n\_qn, n\_sub) :: qn
     INTEGER(kind=1), AllOCATABLE, DIMENSION(n\_scf,n\_sub) :: jcup
     INTEGER(kind=1), ALLOCATABLE, DIMENSION(n\_qn)         :: set
END TYPE CFG
```

Because the parameters differ from one CFG to another, the list would have to be a linked list. This scheme has not been tested and still needs to be verified.

When RCSFGENERATE constructs the list of CFGs, the information about parameters is available. The expansion can be written to `rcsf.out` as a readable files. This file contains the information about `iel, qa, qn, jcup`. The parameter information needed to reconstruct the linked list should be written to a "support" file and, for each CFG, would contain the parameters as well as the values of the `set` array. Thus, it should be possible to reconstruct the linked list as efficiently as possible.

With a linked list structure it is easy to insert new CFG's and to delete CFG's. A question is whether all couplings of a CFG are always needed. For the time being, I assume it will.

The above discussion has assumed that expansions are in block form. I think it is possible to include all J's associated with a CFG but that might require some modification of the above.

3. Selecting CSFs that interact

At this stage, theory could be used to generate the CSFs that interact with a particular reference CSF but at this stage we could use RCSINTERACt (converted to a subroutine) that makes sure the MR set of CSFs precedes all others.

4. The Development process

At an initial stage, a new program RCSFexpand should be developed that

- Includes the MR set at the beginning of each block
- Includes only CSFs that interact with at least one member of the MR set
- produces output in the new format

A preliminary version that outputs in the new format is available. It should be tested and the reordering and "filtering" of CSF for interacation added.

At a subsequent stage, the order in which CSFs are listed should be considered. Calculations are often performed in a systematic manner with orbital sets increasing in size. CSFs could be classified by their "largest" orbital, ie. the orbital with the largest principal quantum number n, and within n the largest angular quantum number l. Then CSF's could be ordered by the nl quantum numbers of these orbitals. Sorting could be sped up considerably with such a scheme. In effect, nl could define a "bin" with sorting not required with such a bin.