ANTOINE MAN

Read, first, A-N generalities concerning the 2 shell model codes ANTOINE (A.) AND NATHAN (N.).

Definition of the basis

Standard calculations (TYPE=0):

```
      DO 1 I=1,2
      READ(IREAD,*) A(I),SH(i),(NLJ(K,I),K=1,SH(i)),(CLAS(K,I),K=1,SH(i)),SAUT(i)
1     CONTINUE
      READ(IREAD,*) MPRO, PARI, JUMP, FLCUT
```

Nocore calculations :

```
      READ(IREAD,*) (A(I),i=1,2), MPRO, PARI, JUMP, FLCUT, JLIM
```

I=1 and 2 for protons and neutrons (or opposite). Especially for large matrices, **take the first space the larger**, larger in the sense of the number of Slater determinants (combinatorial factor of the number of particles and the number of valence states).

If we need 2 different valence spaces (ex: change Tz) we must read successively the 2 valence spaces .

1) the FINAL basis

2) the INITIAl basis

A=number of particles.

SH=number of shells.

NLJ=$1000 * n + 100 * l + 2j$ (definition of the shells)

For Nocore, shells are automatically calculated (1,101,103,205,203,1001,1101,...)

MPRO=$2 * Jz$ PARI= global parity. 0 for positive, 1 for negative.

Notice that

all the quantities which can be half-integer are multiplied by 2.

Truncations.

in Nocore calculations JUMP defines the number of allowed hw excitations . Situation is a little bit more complicated for TYPE=0. At each shell is associated a

number CLAS(m,i) which is used to truncate the space configurations.
We must have always CLAS(m,i) $\leq$ CLAS(m+1,i).

For each configuration, defined by n(k,i) number of particles in each shell k, we calculate :

t(i)=$\sum_k n(k,i) * clas(k,i)$

We define :

tmin(i)=min(t(i)) (all the particles put in the first shells)

and keep only configurations with :

$0 \leq t(i) \leq$ SAUT(i)+tmin(i) in each subspace and

$0 \leq t(1) + t(2) \leq$ JUMP in the global space.

It means that the code takes always SAUT(i)=min(SAUT(i),JUMP).

examples in a specific subspace :

```
 6 4 307 1103 305 1101    0 0 0 0   0      a)
 6 4 307 1103 305 1101    0 1 1 1  10      b)
 6 4 307 1103 305 1101    0 1 1 1   2      c)
10 4 307 1103 305 1101    0 1 1 1   2      d)
```

a) no restriction
b) no restriction, possibility to analyze the wave function in term of t.
c) maximum 2 particles in the space (1103,305,1101)
d) tmin=2 maximum 4 particles in the space (1103,305,1101)

Exception : if we take JUMP < 0 , we take a fixed truncation :
t(1)+t(2) = IABS(JUMP)
********** Limitation JUMP =24

Sharing of the Lancos vectors.

Lanczos vectors are the succession of blocks (rectangular matrices) caracterized by Jz and parity (for each space) We have the possibility to share the vectors following these blocks. the minimum size will be the dimension of the larger block (this quantity is printed in the option calculating the dimensions).

The advantage is that it allows the diagonalization of larger matrices . It gives a natural way to parallelize the code . The disadvantage appears in the pn calculation : we cannot take advantage of the symetry H(I,J)=H(J,I) when I and J do not belong at teh same block .

FLCUT=0 no sharing FLCUT > 0 numero of the file where we read the sharing . FLCUT < 0 useful if you have forgotten the num sharing .

Specificity for Nocore calculations (TYPE > 0) .
A major difficulty when the truncation becomes large is the number of shells (operators) to handle . Following the J value that we want, we can eliminate some shells (example if we calculate J=0 with t=2 in ^{4} He, we know that the shells $0d_{5/2}, 0d_{3/2}$

will not be occupied . These shells will not be eliminated in the basis , but the operators in which they appear will be. To do that take JLIM= maxi of calculated J (if not take JLIM=-1).

V OPTIONS.

- **01. Dimensions of the space.**
- **02. Sharing of the lanczos vectors (average on the length.**
- **03. Sharing of the lanczos vectors (fixed number of blocks.**
- **04. Number of terms in the matrix.**
- **05. three-body .**
- **06 Pivot without projection on J^2 .**
- **07 Idem with projection on J^2 of the Lanczos vectors.**
- **08 Projection on J^2.**
- **11 Lanczos calculation (random pivot).**
- **12 Lanczos calculation (read pivot).**
- **13 Lanczos vectors are projected .**
- **18 Projection on center of mass (nocore).**
- **19 Radii (nocore).**
- **20 Kinetic energy (nocore).**
- **21. Calculation of shells occupations.**
- **22. E(L) and M(L) transitions. Parity unchanged.**
- **23. Sum rule E(L) or M(L). Parity unchanged.**
- **24. Neutrino cross section. Parity unchanged.**
- **31. Change truncation (increase the space).**
- **32. Change Jz .**
- **33. writing vectors for MFD code (nocore) .**
- **34. P-N permutation space .**
- **41. Change the basis .**
- **42. E(L) and M(L) transitions. Parity changed.**

- **43. Sum rule E(L) or M(L). Parity changed.**
- **44. Neutrino cross section. Parity changed.**
- **45. one-body density . Parity changed.**
- **46. Change Tz value.**
- **47. Gamow-Teller sum rule.**
- **48. Gamow-Teller Haxton operator.**
- **49. Gabriel operator . Parity changed.**
- **50. spectroscopic factors.**
- **51.** $0\nu\beta\beta$ **.**

VI SPECIFIC DATA FOLLOWING THE OPTIONS .

01. Dimensions of a valence space

```
     1 0 0 0
6  4 307 1103 305 1101   0 1 1 1     8
10 4 307 1103 305 1101   0 1 1 1     8
  0 0 2 0

     1 0 2 1
 6 5  1 1  8 0
```

Calculation of the dimensions in the space and that for all the Jz values. It gives also the dimension in the J space and the effective dimension of the matrix for M=0 when time reversal symetry is taken in account . It allows to see until what value of J it is better to take Jz=0 instead of Jz=J.
If IPRI=1 calculation of dimensions for JUMP=0,JUMP
If IPRI=2 calculation of dimensions for JUMP=mod(JUMP,2),JUMP,2
(in the second example we calculate dimension for t=0,2,4,6)

02. Sharing of the vectors (with Dimension)

```
     2 0 0  0
10 4 307 1103 305 1101      0 1 1 1     8
12 4 1103 305 1101 409      0 0 0 1    10
0 0 18 70
  400                                        a)
```

a) MOY size that we try to get in average .

03. Sharing of the vectors (in N blocks)

```
     3 0 0  0
10 4 307 1103 305 1101      0 1 1 1     8
12 4 1103 305 1101 409      0 0 0 1  10
0 0 18 70
  7                                    a)
```

a) N number of blocks that we need (option useful for parallelization N=number of processors !)

04. Number of non-zero of the matrix

```
 4 0 0 0
10 4 307 1103 305 1101       0 1 1 1  8
12 4     1103 305 1101 409   0 0 0 1 10
 0 0 18 70
```

Number of non-zero in the matrix . The result is given for each block (Jz,parite) and if FLCUT > 0 for all the sghared components.
This option gives also the number of equivalent integer numbers to store on each disk.

06. Creation of pivot(s) without J^2 projection.

```
 6 0 0 1
 50 0 0
 6 4   0 0   4 0  -1
 2  1 -1 245789  1            a)
```

a) NCAL, (RVT(k),k=1,NCAL), KPVT, RVI
RVT : if MPRO (Jz) =0 we must define a time reversal symetry :
+1 to have a pivot containing even J values
-1 to have a pivot containing odd J values
if MPRO $\neq 0$ RVT=0 (it means that NCAL must be =1).

The created pivots are random . KPVT is an integer which initiates the random numbers.

RVI=1 pivot states are random but built with T=Tz.
RVI=0 no restriction on isospin .

Supposing A(1) $\geq$ A(2), states of the basis for which all the occupied states in space 2 are occupied in space 1 have T=Tz. These states allow to get a pivot state with good isospin . (It is not possible to get it for a nucleus N=Z=odd with Jz=0) .

07. Creation of pivots projected on J^2 .

```
 7 0 0 0
50 0 0
 4 4 307 1103 305 1101       0 0 0 0  0
 2 4 307 1103 305 1101       0 0 0 0  0
 0 0 0 0
3 0 2 4                                  a)
```

b) NCAL,(JVAL(k),k=1,NCAL)
JVAL=2*J of the calculated states.
note: the code calculates and stores ,first, the states having same time reversal than the first one . In the present example we will have on fiel 50 the states J=0,2 and after J=1.

08. Projection on J^2.

```
     8 0 0  0
 51 50 0 0 2                                  a1)
6 4 307 1103 305 1101      0 1 1 1     8
4 4 307 1103 305 1101      0 1 1 1     8
0 0 2 0
  2  0  4              b)
  1   6               b)

  8 0 0 0
 0  50 0 0 0                                  a2)
6 4 307 1103 305 1101      0 1 1 1     8
4 4 307 1103 305 1101      0 1 1 1     8
0 0 2 0
```

a) FLWRI, FLLEC, (NLEC(k),k=1,2), NCAL
2 possibilities:
a1) FLWRI $\neq$0 , reading of :

b) NKEEP, (JVAL(k),k=1,NKEEP) and that for all the NCAL states. . If NCAL=0 , it is taken NCAL=1 . If NCAL is larger than the number of vectors on FLLEC, the code stops .

a2) FLWRI 0 . No reading of b) . If NCAL=0 , all the vectors of the file FLLEC are considered . This is the main interest of this option, allowing to check the angular momentum of vectors .

11. Lanczos calculation (random pivot).

```
11 1 0 0                                     a)
Ti46-t2                                      a)
20 0 0                                       b)
 4 4 307 1103 305 1101      0 1 1 1  2
 2 4 307 1103 305 1101      0 1 1 1  2
 0 0 2 0
    93 0                                  c0)
```

```
  3  0 2 4                                   d)
     1 1 2                                   e)
  45 0.0005  0 0                             f)

11 1 0 0                                     a)
Ti46-t2                                      a)
20 0 4                                       b)
 4 4 307 1103 305 1101      0 1 1 1  2
 2 4 307 1103 305 1101      0 1 1 1  2
 8 0 2 0
    93 0                                   c0)
  2  8 12                                   d)
     1  1                                   e)
  45 0.0005  0 0                             f)

11 0 0 0                                     a)
21 0 0                                       b)
 4 4  0 0  0 0  -1
    80 2  10.                              c1)
  3  0 2 4                                   d)
     1 1 2                                   e)
  45 0.0005  0 0                             f)
```

a,b) see generalities

Notice that in b1) we have STOR=4 . It means that 4 states will be read and will appear at the end in the spectrum.

c0) FLNUC,COUL standard SM

FLNUC= file of H

COUL= 0 no coulomb ; COUL=1,2 proton space (coulomb must be included in H)

c1) FLNUC,PPPP, ZCDM Nocore

PPPP=1,2 proton space . ZCDM= multiplicative factor for Hcm.

d) NCAL, (JVAL(k),k=1,NCAL) states to calculated.

remember the order of the states when MPRO=0 (time reversal symmetry). In the present example, we will get the J=0 state, then the two J=2 states and at the end the J=1 state. This is very important to remember , especially if these states will become pivots in a larger spase (see next option) .

e) (NCON(k),k=1,NCAL) number of converged states that we need for each matrix.

f) NLOOP, ZFIT, IORTH, NLANZ

NLOOP= number maximum of Lanczos iterations

ZFIT : test of convergence.

each 3 iterations the tridiagonal Lanczos matrix is diagonalized. If all the states that

we want to converge have an increase of the energy smaller than ZFIT, the Lanczos procedure is stopped.

For standard calculation take ORTH=NLANZ=0 (see later specific possibilities)

12. Lanczos calculation (read pivot).

Usually in this option, pivot states are the eigenvectors of a smaller space.

```
12 0 0 1
21 10 0 0 0                          a
 4 4  0 0  6 0   4                   d)
    81 2  10.
2   1 1                              b)
0 45 0.0005  0 0                     c)
```

Same reading that the precedent option, except:
a) we have 2 files, the pivot states are on FILV(2)
The reading of JVAL(k) has desappeared.

b) NCAL, (NCON(k),k=1,NCAL) same meaning precedent option. c) KPVT, NLOOP, ZFIT, IORTH, NLANZ
In this line a new reading :
KPVT $>$ 0 pivot : linear combination of the KPVT vectors on the initial file.
KPVT = 0 pivot : linear combination of the NCON(k) vectors on the initial file.

13. idem with projection of Lanczos vectors on J^2.

```
 13 1 0 0
 Cr50.t2
 50 51 0 0 0
 1 1 0                                              a)
6 4 307 1103 305 1101      0 1 1 1     8
4 4 307 1103 305 1101      0 1 1 1     8
0 0 2 0
    90 0
 2  3 3
  1   30  0.0005  0  0
```

a) (PRV(k),k=0,2)

When many Lanczos iterations are done (many converged states are needed, ...), numerical errors appear; symetry of H can be lost . A usual example : you calulate many J=2 states and at the end you get A j=2 state degenerated with the J=0 ground state. You haverecalculated the GS . Notice that the "catastrophe" is very abrupt and can be easily detected in looking at the energy convergence.

To avoid this problem, we can project each new Lanczos vectors on :
PRV(0)=1 time reversal symetry (not necessary for odd-odd nuclei for which it is calways conserved).
PRV(1)=1 Angular momentum
PRV(2)=1 center of mass (only in nocore calculations)

Notice that this procedure is "expansive", not totally secure, not implemented with shared vectors and not to be used with simple precision calculations . In some situations an alternative can be the orthogonalization (see later) .

Special possibilities with Lanczos.

STRENGTH FUNCTIONS

```
 13   0  0
 50 51 0 0 0
 1 1 0
6 4 307 1103 305 1101      0 1 1 1     8
4 4 307 1103 305 1101      0 1 1 1     8
0 0 2 0
    90 0
  1    0
  1  100 -0.0005  0  0                        a)
    20  2  1                                  b)
```

a) In a strength function we calculate Lanczos vectors until the end without convergence criteria . For that take ZFIT < 0.0

b) STEP, (NFOR(k), NOVER(k),k=1,NCAL)
In standart Lanczos we have STEP=3 . If you do many iterations take a larger value .

The NFOR(k) states, higher in energy than the NCON first states, and having a maximum overlap with th pivot state are kept in FIL(1). At the end we kept always NCON+NKEEP vectors.

NOVER(k) is the number of vectors which "follows" the pivot on FIL(2) and for which a strength function will be calculated (with the Lanczos vectors precedently calculated). Notice that for these states the total strength will not be 1.

RESTART A LANCZOS CALCULATION.

```
12 0 0 0
.................
 1   30  0.0005  0 100       a)

a) For different reasons you can want to restart a Lanczos calculation
   ( increase the number of converged state, improve a strength function, ...)
  Give a value at NLANZ . The code checks the number NSTOR of vectors
```

```
   that he has and restart Lanczos with NLANZ = min ( NLANZ, NSTOR).
   ( at each iteration , the code writes in a file called "tridia" the
   tridiagonal matrix . Notice that NLANZ is

\vspace*{2mm}
  ORTHOGONALIZATION TO THE PRECEDENT CALCULATED STATES

\begin{verbatim}
 12   0  0
 50 51  2 0 3
 ................
  2  40 80
      3  3
  1   30  0.0005  1   0       a)
```

a) ORTH=1 :
we orthogonalize each new Lanczos vector to the vectors precedently caculated .In the present example teh 2 fist vectors on file 50 will be forgotten, the orthonlization will be done with the 3 following vectors for state 40 and the 3 +2 states 40 for state 80 . Vectors with differnt dimensions, ... are forgotten.

It can be an alternative at the projection of Lanczos vectors (option 13), especially when the vectors ares shared. It can be used to calculate a new state of a given spin , or converge succesively 2 states instead of the 2 both.

18 Projection on center of mass (nocore). 18. Projection on center of mass (nocore) .

```
 18 0 0 2
 50 51 0 0 2              a)
 2 2   0 0   12 0  -1
```

a) like always if NCAL=0 (here insrtead of 2) all the vectors on FILV(2) are projected .

We keep only the Os cdm state.

19 Radii (nocore).

```
 119 0 0 2
 51 0 0
 4 2   0 0   16 0   0
```

same comment concerning NCAL . In this example, vectors are in simple precision (CAS=119 instead of 19) .

20 Kinetic energy (nocore).

data idem calculation of radii.

21. Occupation of Shells

```
 21 0 0 0
 50 0 0                                              a)
6 4 307 1103 305 1101       0 1 1 1      8
4 4 307 1103 305 1101       0 1 1 1      8
0 0 2  0
```

a) Remember that if NCAL=0 all the vectors of the file are read

Results of the options are :

Occupation of shells. Component for each truncation

For stabndard calculation, print the configurations if the occupation is > 0.01 .This number can be changed (see subroutine TCONFA in tconf.f) .

21. Transitions (no parity change).

```
 22 0 0 0
 50 0 0
6 4 307 1103 305 1101       0 1 1 1      8
4 4 307 1103 305 1101       0 1 1 1      8
0 0 2  0
2 0 20 20 0.5 1.5         a)

 22 0 0 0
 50 0 0
6 4 307 1103 305 1101       0 1 1 1      8
4 4 307 1103 305 1101       0 1 1 1      8
0 0 2  0
 1 0  20 20    -3.826   0.000   5.586 1.0     a)

 22 0 0 1
 50 0 0
6 4   0 0      4 0 -1
2 0  12.0 0.0 1.0         b)
```

a) LB,DP,(CORE(k),k=1,2),(ZCOEF(K,PN),K=1,1+MOD(LB+DP,2)),PN=1,2)

b) LB,DP, ZHW ,(ZCOEF(K,PN),K=1,1+MOD(LB+DP,2)),PN=1,2) LB=lambda DP=0 no parity change (in this option)

b) CORE : number of particles in the core for space 1 and 2 . (used only to calculate the oscillator length.

c)= In the nocore calcations ZHW is the hw used in the interaction.
ZCOEF=
electric transition : effective charge 1 and 2
magnetic transition : sigma(1), orbital (1), sigma(2) , orbital (2)

23. Sum Rule (no parity change).

```
 23 0 0 0
 51 50 0 0 1
4 4 307 1103 305 1101       0 0 0 0     8
3 4 307 1103 305 1101       0 0 0 0     8
1 1 0  0
2 0 20 20 0.5 1.5
```

data idem precedent option, except tha we have 2 files. If the initial is not J=0, the sumrule is projected on angular momentum.

24. Neutrino cross section (no parity change).

```
 24 0 0 0
 51 50 0 0 1
2 2   0 0  10 0  -1
 ..........                    a)
 ..........                    a)
```

a) calculation of sum rules for different operators. See elneu.f

31. Change truncation.

```
 31 0 0 0
 51 50 0 0 0
6 4 307 1103 305 1101       0 1 1 1     8
4 4 307 1103 305 1101       0 1 1 1     8
0 0 6 0                                          a)
```

a) We read the final truncation . Read vectors (on FIL(2)) must have the same type of truncation but smaller or equal.

32. Change Jz .

```
 31 0 0 0
 51 50 0 0 0
6 4 307 1103 305 1101       0 1 1 1     8
4 4 307 1103 305 1101       0 1 1 1     8
2 0 6 0
```

33. **write data to use in MFD code (nocore).**

```
 32 0 0 0
 51 50 0 0 3
6 4  0 0 6 0 -1
```

34. **permutation of p and n space.**

```
 34 0 0 0
 51 50 0 0 0
6 4 307 1103 305 1101       0 0 0 0      0
2 4 307 1103 305 1101       0 0 0 0      0
0 0 0 0
```

we obtain the solution corresponding at the permutation of the 2 supspaces.

In all the option with CAS > 40 , we have different basis for the ifinal and the initial basis . We read always firstly the final basis and after the initial one . The only condition is that the shells must be identical. **41.** **Change**

The Basis .

```
41 0 0 0                                                    a)
63 50 0 0 1
  4  4 307 1103 1101 305      0 1 1 1         8
  2  4 307 1103 1101 305      0 1 1 1         8
 0 0 -2 0
  4  4 307 1103 1101 305      0 0 0 0         8
  2  4 307 1103 1101 305      0 0 0 0         8
 0 0  0

41 0 0 0                                                    b)
63 50 0 0 1
  4  4 307 1103 1101 305      0 0 0 0         8
  4  4 307 1103 1101 305      0 0 0 0         8
 0 0  0 0
  4  4 307 1103 1101 305      0 0 1 1         8
  4  4 307 1103 1101 305      0 0 1 1         8
 0 0  0 0
```

the truncation can be totally different . We can increase the space, reduce it or mix the 2 things . a) In this example we extract from a calculation without

truncation, the component with a fixed truncation.
b) If you want to add shells in the valence space you can "play" with the truncations . In this example we write vectors from the restricted (f7/2,p3/2) space in the full space.

42. Transitions (parity change).

```
 42 0 0 0
 50 0 0                                  a)
4 4 1103 305 1101  409 0 0 0 0  0
4 4 1103 305 1101  409 0 0 0 0  0
0 0 0  0                                b)
4 4 1103 305 1101  409 0 0 0 0  0
4 4 1103 305 1101  409 0 0 0 0  0
0 1 0  0                                b)
1 1 20 20 0.5 1.5        c)

a) All the vectors must be on the same file.

b) In this example we calculate transitions from
  negative parity states (second basis) to positive.

c)  reading idem no parity change. However notice
 that for E1 transition the read coefficients are irrelevant.

\vspace*{2mm}
       {\bf 43. \hspace*{3cm} Sum rule    ( parity change).}
\vspace*{2mm}

\begin{verbatim}
 43 0 0 0
 51 50 0 0 1
4 4 1103 305 1101  409 0 0 0 0  0
4 4 1103 305 1101  409 0 0 0 0  0
0 1 0  0
4 4 1103 305 1101  409 0 0 0 0  0
4 4 1103 305 1101  409 0 0 0 0  0
0 0 0  0
1 1 20 20 0.5 1.5

 initial state in the second basis (Here positive parity state).
```

```
      {\bf 42. \hspace*{3cm} Change Tz.}

\vspace*{2mm}

\begin{verbatim}
15 0 0 0
63 50 0 0 2
 3   4 307 1103 1101 305      0 0 0 0         0      a)
 3   4 307 1103 1101 305      0 0 0 0         0
 0 0 0 0
 4   4 307 1103 1101 305      0 0 0 0         0      b)
 2   4 307 1103 1101 305      0 0 0 0         0
 0 0 0 0
```

we define the final basis a) and the initial b)

16. Gamow-Teller.

```
16 0 1
0  50 0 0 1
  5  4 307 1103 1101 305      0 1 1 1         8      a)
  1  4 307 1103 1101 305      0 1 1 1         8
 0 0  3
  4  4 307 1103 1101 305      0 1 1 1         8      b)
  2  4 307 1103 1101 305      0 1 1 1         8
 0 0  2
```

we define the final basis a) and the initial b) If the initial state is J=0 , the final state is J=1 and it is written on the file FIL(1) (if FIL(1) $\neq$ 0)

If (second example) the initial state is not J=0 we must project this "sum rule" state on J^2.

Note :

Possiblity to have JUMP final $\neq$ initial (in the present example to keep Ikeda sum rule)

In the second example we have Jz=0 excluding to have a final state J=2 since J initial=2 . To have it Jz value must be changed .(see next option)