ANTOINE-NATHAN.MAN

In this manual we give generalities conrening the shell model codes ANTOINE (A.) AND NATHAN (N.).

These codes allow the diagonalization of shell model wavefunctions.
In A. they are written in M scheme.
In N. they are written in a coupled (J but not T) basis.

They have the possibility to calculate energies, wave-functions, transitions, sum rules,....

They include tools useful for shell model calculations (modification of two-body matrix elements...)

All these possibilities are defined by a numero of an option and they can be executed successivelly.

The codes are (yet !!) in perpetual evolution :

Add new options, improve performances of some of them ... We never know the influence of some modification ... which explain that THERE IS NO GUARANTEE on the exactness of the results of ALL the options with ALL the possible data . Lot of tests have beee done, but we never know ...

However if the reults are wrong or if the code breaks, check the data and the total dimension allowed for the calculation before to suspect the code and after that send me the data and the results. Thanks in advance.

Etienne CAURIER
tel 03-88-10-64-84
email etienne.caurier@ires.in2p3.fr

Generalities

The diagonalization in itself is done with the Lanczos method.

The problem is to avoid the precalculation and storage of all the non-zero of the matrices. The fundamental idea of the codes is the factorization of the basis in 2 spaces. Precalculation of operators are done (and stored) for each subspace . This allows a fast generation of the non-zero of the matrices (3 integer additions for A. ; 3 integer additions + 2 floating multiplications for N.). the 2 subspaces are proton and neutron spaces . However in N. it is possible to put in the p (n) subspace some n (p) shells .The goal is to balance the 2 spaces, which reduces the prestorage . This mixing is not trivial to choose (see in N.man).

Details concerning the theory of these codes can be found in the following references :

- Present Status of Shell Model Techniques, Zacopane School Septembre 1998, Acta Phys. Polonica **30** (1999) 705.
- Systematics of binding energies from full 0hw pf shell model calculations, Phys. Rev. **C59** (1999) 2033.
- Frontier in Shell Model Calculations (Tokyo Conference March 2000), Nucl. Phys. **A704** (2002) 60c
- Advances in Shell Model Calculations (Maiori Conference, June 2001)

The code is written in FORTRAN NORM 77

Specificities of the programmation :

except logical and character type variables, All the quantities beginning with letters :

A-U are integers; V-Z are double precision.

All the data (except character type) are read with a free format.

Commons are not aligned. Furthermore a same common can be used in different subroutines to store different things (this is the case for all the commons beginning with AAA).

2 possible main programs :

aaa9.f with "malloc" to get larger memory.

aab9.f standard but also used with 64 bits processor.

In the main program we define the fortran units for input (IREAD) and output (IOUT) and the parameter TOT, length of the array F .

In F(∗) are written all the most important arrays needed in the calculation. Notice that even if F is defined integer, it can contain double precision quantities .Limitations :

TOT $\leq$ RAM + swap and TOT being an integer ∗4 TOT $\leq 2^{31} \simeq 2 * 10^9$

During a Lanczos calculation, F contains 2 vectors (or 2 pieces of these vectors) + some working lengths. During the calculation the code tests often if the size is sufficient (if not, it stops with a clear message). But the tests are not always possible... so sometimes it can break ... Before to suspect an error in the code, increase this parameter.

The first subroutine (acas.f) fixes some quantities (LNN,LW2, ...) . The code checks them and send a message if it is necessary to increase them . Some dimensions are fixed in some commons. Some dimensions are arbitrary fixed. During the execution of the job, they are (in principle !) tested and a clear message appears if they are too small.

II. Files

All the precalculations, which are done separetely in the two spaces are stored. This storage (especially for large truncation in nocore calculations) can be huge . For this reason we define in "acas.f" NDISK=k and the storage will be equilibrated between the directories "amat1, amat2, ..., amatk".
NDISK$\leq$5 (see opfil.f)

The Lanczos vectors are stored in a directory called "vec".

The Hamiltonian and the vectors files are defined with a numero which is read in the data and corresponds to a Fortan unit .
ln -sf $HOME/fp/Ti46fort.50$
$ln - sf$HOME/hamil/kb3 fort.90
ln -sf /pcs31d1/caurier/TBME/A6/TBMEA6INOY10.12bin fort.24

If a file is not used take numero 0. For example if you want only energies without eigenvectors, take 0 for this file.

Fortran Units 1,2,3,4 are reserved for temporary files in "amatk" and "vec" .

At the end of an option, files are in principle rewinded. It is possible to avoid that (for example you read a vector in an option and you want to write an other one behind in the next option) For that take a negative value for the numero of the file (absolute value will be taken in the calculation)

III Interactions

For Nocore calculations, interactions are on binary files calculated by P. Navratil .

Interactions Files for standard calculations :

```
 USD INT.                                          a)
1 3 203 205 1001                                   b)
  1.64658  -3.94780  -3.16354                      c)
   1    8    8    0.300000    0.000000            d)
0 1 203 203 203 203 0 3                            e)
  0.00000  -1.41510   0.00000  -2.88420            f)
 -2.18450   0.00000  -0.06650   0.00000
0 1 205 203 203 203 1 3
  0.56470   0.00000   2.03370
  0.00000  -0.61490   0.00000
 ................................
0 1 1001 205 1001 203 2 2
  2.06640
 -1.94100
 ..........................
```

```
 0 1 1001 1001 1001 1001 0 1
   0.00000  -3.26280
  -2.12460   0.00000
```

a) TEXT (character∗30)

This is the name of the interaction. This will be written with eigenvectors of the diagonalization. (after some months it can be useful to remember what interaction has been used !)

b) QMAX, SH, (NLJ(k), k= 1, SH)

QMAX=1 individual energy identical for a proton or neutron shell
QMAX=2 individual energy different for a proton or neutron shell
SH= number of shells in the valence space
NLJ= definition of the shell nlj with NLJ= $1000 * n + 100 * l + 2 * j$

n begins at 0 . Examples 307 = 0f7/2 1103 = 1p3/2 . 511 = 0h11/2 .

This notation for the shells will be the same in all the data.

C) Loop on q=1,QMAX
(ZEPSI(k,q), k= 1, SH)

ZEPSI=individual energies. If QMAX=2 q=1=neutron q=2=proton

D) IDENS, (CORE(k),k=1,2), ZMASS

density dependance IDENS=0 no dependance with the mass ; =1 dependance fot only TBME ; =2 dependance for TBME and individual energies.
CORE = number of particles (space 1 and 2) in the core.
the multiplicative factor is :
XMULT=(AINI/ATOT)**ZMASS with AINI=CORE(1)+CORE(2)+2.

(notice that a second coefficient is included in the file, allowing a dependance with isospin) .

e) TMIN, TMAX, (NLJ(k), k= 1, 4), JMIN, JMAX

f) loop t= TMIN, TMAX
(VMAT(j, t), j= JMIN, JMAX)

t=isospin. For standart Hamiltonian take TMIN=0 TMAX=1
If V(pp) $\neq$ V(nn) take TMAX=2 V(pn,T=1)=V(nn)=V(t=1) and V(pp)=V(t=2)
If V(pp) $\neq$ V(nn) $\neq$ V(pn,T=1) take TMAX=3 V(pn,T=1)=V(t=1); V(nn)=V(t=2) and V(pp)=V(t=3)
(it is always possible to have TMIN $\neq$ 0.

If the spaces for protons and neutrons are different,we can have matrix elements defined in p-n formalism.In that case we will have

TMIN=1 for V(pp) and V(nn)
TMIN=-1 for V(pn) (loop on t=iabs(TMIN),TMAX)

And in these cases we read only (X(j), j= JMIN, JMAX)

important note:

The order of shells has no importance. The code checks that all the individual energies are defined (if not, it stops with a message), but does not check if some matrix elements are missing , they will be put at a value 0.0 without a message . However the total number of readings e) is printed .

GENERALITIES about DATA

We give here the fundamendal readings which occur often in the different options. Examples for each option with explanation about their specific data are given later.

All the data (except character type data) are read with a free format. All the Fortran variables beginning with a letter :

A-U are integers

V-Z are double precision.

First reading is the same for ALL the options.

```
      READ (IREAD,*,END=99) CAS, ITEXT, IPRI, TYPE
      IF(ITEXT.NE.0) READ (IREAD,100,END=99) TEXT
100   FORMAT(A30)
```

CAS= numero of the option (see list).
If CAS=0 or has not been defined, it stops.

TEXT is a variable of character-type which will be written with the the stored vectors (or Hamiltonian).

IPRI=0 standard, IPRI=1,2 more intermediate printings .

for A. we have :
TYPE=0 Standard valence space .
TYPE=1 No Core valence space .
TYPE=2 No Core maxi (N,Z) $\leq$ 2 .
TYPE=2 calculations can be done with TYPE=1 if truncation is not too large.

in N. TYPE (called IPARA !) is related at possible parallelization (see later).

After this reading, except the options concerning manipulations of files or interactions (CAS $>$ 60) which have specific data, all the others have the following reading :

```
       IF(KF.GT.0)
      *READ(IREAD,*) (FILV( k ), k= 1,KF),(INIV( k ), k= 1,KF), STOR
```

KF, defined with a data in the subroutine achoi.f , is the number of files of vectors that you can have to manipulate during the option. It can be

1. KF=0 we calculate dimensions ...
2. KF=1 we read OR write vectors (calculation of transitions ...).
3. KF=2 we write AND read vectors (usual Lanczos calculations ...) .

FILV(k) is the numero of the file (can be =0 if not used)

If KF=2 :

FILV(**1**) is always the file on which we **write** vectors.

FILV(**2**) is always the file on which we **read** vectors.

Remember that at the end of the option, the files are rewinded, except if the numero of the file has been defined negative.

INIV(k) (initialisation of the files) is the number of vectors read at the beginning of the task.

If INIV(k) is larger than the number of stored states, a message is given, a backspace is done and the program continues.

STOR : 2 possible meanings following the options .

When we calculate new vectors , it is the number of vectors precedently calculated . They will be taken in account in the calculations of spectra, and also if orthoganlization is needed. At the opposite the vectors INIV are completly forgotten .

When we do operations on vectors (calculation of transitions ,sum rule,....) it is the number of vectors which will be treated . In this case STOR is often called NCAL. STOR=0, all the vectors on the file will be taken in account (STOR is transformed in 10000) . It avoids to know precisely the number of vectors on the file .