TAURUS$_{\text{vap}}$: Manual for the input file

05/01/2021

1 Structure of the input file

The input file is read by the code as STDIN and therefore has no fixed naming convention. On the other hand, the format of the file is fixed. We describe in this manual how to write a proper input file for TAURUS$_{\text{vap}}$ and the different options that the code offers. Because the input file is somewhat lenghty, we will present its different section separately (but remember that they are all part of the same file).
Before going further, a few remark are in order:

- We use here the Fortran convention for the format of variables, e.g. 1i5 or 1a30, assuming that the reader know their meaning. If it is not the case, we encourage the reader to search for a tutorial somewhere else. There is a large body of Fortran documentation available online therefore it should not be too difficult.
- All lines starts with the reading of an element of an array, `input_names` or `input_block`, made of `character(30)` variables. These variables only play a cosmetic/descriptive role when reading or printing the input parameters and can be set to any desired value by the user. Therefore, we are not going to comment them further.
- The number of lines in the input files depends on the number of quasi-particle one wants to block. Here, we will give the line numbers considering an input without any blocking.

1.1 Section about the Hamiltonian

Description

Line	Format	Data
1	1a	`input_names(1)`
2	1a	`input_names(2)`
3	1a30, 1a100	`input_names(3), hamil_file`
4	1a30, 1i1	`input_names(4), hamil_com`
5	1a30, 1i1	`input_names(5), hamil_read`
6	1a30, 1i5	`input_names(6), paral_teamssize`
7	1a	`input_names(7)`

where

- `hamil_file`: common name of the Hamiltonian input files. It is used by the code to determine all the possible hamiltonian files: `hamil_file.sho`, `hamil_file.01b`, `hamil_file.2b`, `hamil_file.com`, `hamil_file.red`
- `hamil_com`: option to take into account the center-of-mass correction when doing calculations with general Hamiltonians (`hamil_type = 3 or 4`).
  `= 0` w/o center-of-mass correction.
  `= 1` with center-of-mass correction, whose two-body matrix elements are read from `hamil_file.com`.
- `hamil_read`: option to read the Hamiltonian from the normal files or from a "reduced" file.
  `= 0` reads the Hamiltonian from the normal Hamiltonian files (`hamil_file.sho`, `hamil_file.01b`,

hamil_file.2b, hamil_file.com)

= 1 reads the Hamiltonian from the reduced file hamil_file.red containing the m-scheme matrix elements written in binary (to be faster and have a smaller size). Note that the file hamil_file.sho is still needed and read by the code. By contrast, hamil_file.com will not be read even if hamil_com = 1, so the center-of-mass correction has to be already included in the reduced file.

- paral_teamssize: number of MPI processes in a team. Within a team, each process stores only a certain subset of the two-body matrix elements of the Hamiltonian and computes with them the fields h, Γ and Δ. Then they all synchronize through a mpi_reduce. This option is needed for large model spaces when the storage of the two-body matrix elements in the SHO basis becomes a problem.

  = 0 or 1 all processes have access to all matrix elements.

  > 1 the number of matrix elements managed by a process, hamil_H2dim, is obtained by performing the euclidean division of the total number of matrix elements, hamil_H2dim_all, by the number of processes within the team, paral_myteamsize. The rest of the division is distributed among the first processes in the team (i.e. the processes with a rank in the team, paral_myteamrank, strictly lower than the rest). Note that if paral_teamssize is not a divisor of the total number of MPI processes, paral_worldsize, then the last team will only be made of modulo(paral_worldsize,paral_teamssize) processes. Therefore, the actual size of the team is not necessarily equal to the input paral_teamssize. Obviously, paral_teamssize has to be lower than or equal to paral_worldsize.

Example

```
Interaction
-----------
Master name hamil. files     usdb
Center-of-mass correction    0
Read reduced hamiltonian     0
No.  of MPI proc per H team  0
```

1.2 Section about the particle number

Description

Line	Format	Data
8	1a	input_names(8)
9	1a	input_names(9)
10	1a30, 1f7.2	input_names(10), valence_Z
11	1a30, 1f7.2	input_names(11), valence_N
12	1a30, 1i5	input_names(12), proj_Mphip
13	1a30, 1i5	input_names(13), proj_Mphin
14	1a	input_names(14)

where

- `valence_Z`: number of active protons, either on average (HFB) or exact (VAPNP). For no-core calculations, this is the total number of protons in the nucleus. Note that for VAPNP calculations, this number has to be an "integer" (e.g. `4.00`).
- `valence_N`: same for the number of active neutrons.
- `proj_Mphip`: number of points in the discretization (à la Fomenko) of the integral over proton gauge angles.
  `= 0` no particle-number projection is performed for this particle species. Note that this is equal to the case `proj_Mphip = 1`.
  `> 0 and odd` use a Fomenko discretization with the n-th point being located at the angle $(n-1)/($`proj_Mphip`$)$.
  `> 0 and even` use a Fomenko discretization with the n-th point being located at the angle $(n-1/2)/($`proj_Mphip`$)$
- `proj_Mphin`: same for the integral over neutron gauge angles.

Example

```
Particle Number
---------------
Number of active protons      4.00
Number of active neutrons     5.00
No. of gauge angles protons   5
No. of gauge angles neutrons  5
```

1.3 Section about the wave function

Description

Line	Format	Data
15	1a	`input_names(15)`
16	1a	`input_names(16)`
17	1a30, 1i1	`input_names(17), seed_type`
18	1a30, 1i5	`input_names(18), blocking_dim`
		`do i=1, blocking_dim`
	1a30, 1i5	`  input_block(i), blocking_id(i)`
		`enddo`
19	1a30, 1i1	`input_names(19), seed_symm`
20	1a30, 1i5	`input_names(20), seed_rand`
21	1a30, 1i1	`input_names(21), seed_text`
22	1a30, 1es10.3	`input_names(22), seed_occeps`
23	1a30, 1i1	`input_names(23), seed_allemp`
24	1a	`input_names(24)`

where

- `seed_type`: option to select the type of seed used as initial wave function.
  `= 0` random real general quasi-particle state.
  `= 1` read from the file `initial_wf.bin/txt`.

= 2 random spherical BCS state (with good parity, good angular momentum and w/o proton-neutron mixing).

= 3 random axial BCS state (with good parity, good third component of the angular momentum and w/o proton-neutron mixing).

= 4 random general quasi-particle state with good parity.

= 5 random general quasi-particle state w/o proton-neutron mixing.

= 6 random general quasi-particle state with good parity and w/o proton-neutron mixing.

= 7 random Slater determinant.

= 8 random Slater determinant with good parity.

= 9 random Slater determiant built from HO single-particle states (with good parity and good third component of the angular momentum).

- `blocking_dim`: number of quasi-particle to block before the iterative procedure.

- `blocking_id(i)`: index of the i-th quasi-particle to block. Note that each quasi-particle index is read on a different line and that no index is read if `blocking_dim = 0`.

- `seed_symm`: option to switch on/off the eventual simplifications of the particle-number projection resulting from the symmetries of the initial wave function.

  = 0 the simplifications are performed if possible: i) `proj_Mphip` and `proj_Mphip` can be set to 1 if the initial wave function has a good number of protons and neutrons, respectively, and ii) the integration over gauge angles can be reduced to $[0, \pi]$ if there is no mixing between protons and neutrons. The code will also checks that there is no symmetry-breaking constraint before doing the simplifications.

  = 1 no simplification is performed.

- `seed_rand`: option to manually set the seed of the random number generation.

  = 0 the seed is generated using the state of the processor during the execution. The seed will be different in each run.

  > 0 the seed is generated using the input parameter and a small deterministic algorithm. The seed may still be different when using different compilers or processors.

- `seed_text`: option to read/write the initial/final wave function as a binary (.bin) or text (.txt) file.

  = 0 reads the file `initial_wf.bin` and writes the file `final_wf.bin`.

  = 1 reads the file `initial_wf.txt` and writes the file `final_wf.txt`.

  = 2 reads the file `initial_wf.bin` and writes the file `final_wf.txt`.

  = 3 reads the file `initial_wf.txt` and writes the file `final_wf.bin`.

- `seed_occeps`: cutoff to determine the fully occupied/empty single-particle states in the canonical basis. This is needed to compute the overlap using the Pfaffian method. We recommend changing this parameter only when observing serious numerical problems or inaccuracies.

  = 0.0 the code will assume the value `seed_occeps` = 10^{-8}.

  > 0.0 the code will consider single-particles states with $v^2 \geq 1.0 -$ `seed_occeps` as fully occupied and the ones with $v^2 \leq$ `seed_occeps` as fully empty.

- `seed_allemp`: option to take into account the fully empty single-particle states in the calculation of the overlap. As for the previous option, we recommend to switch on this option only if you encounter numerical problems or if the code crashes with an error messsage linked to the

evaluation of the overlap.

`= 0` eliminates the maximum number of empty states possible.

`= 1` takes into account all the empty states.

Example

```
Wave Function
-------------
Type of seed wave function    0
Number of QP to block         1
Index QP to block             13
No symmetry simplifications   0
Read/write wf file as text    0
Cutoff occupied s.-p. states  0.000E+00
Include all empty sp states   0
```

1.4 Section about the iterative procedure

Description

Line	Format	Data
25	1a	`input_names(25)`
26	1a	`input_names(26)`
27	1a30, 1i5	`input_names(27), iter_max`
28	1a30, 1i5	`input_names(28), iter_write`
29	1a30, 1i1	`input_names(29), iter_print`
30	1a30, 1i1	`input_names(30), gradient_type`
31	1a30, 1es10.3	`input_names(31), gradient_eta`
32	1a30, 1es10.3	`input_names(32), gradient_mu`
33	1a30, 1es10.3	`input_names(33), gradient_eps`
34	1a	`input_names(34)`

where

- `iter_max`: maximum number of iterations in the minimization procedure.
  `= 0` no iterations are performed but the wave function is still read/generated and the constraints adjusted. The code still performs the final printing of the properties (with and w/o projection).
  `> 0` perfoms at maxmimum `iter_max` iterations, and less if the calculation is converged.

- `iter_write`: number of iterations separating two intermediate writing of the wave function.
  `= 0` the code does not perform any intermediate writing of the wave function.
  `> 0` the codes writes every `iter_write` iterations the wave function of the current iteration to the file `intermediate_wf.bin/txt` (the file follows the same rule as `final_wf.bin/txt`).

- `iter_print`: option to ask more intermediate printing at each iteration (requires more CPU time).
  `= 0` prints the minimum amout of informations at each iterations
  `= 1` prints in addition the expectation value of the parity and average triaxial deformations.

- `gradient_type`: option to select the weight factors in the gradient+momentum algorithm, with the gradient at iteration i being defined as $G(i) = \eta(i)H^{20}(i) + \mu(i)G(i-1)$.
  `= 0` The weights are fixed with $\eta(i) =$ `gradient_eta` and $\mu(i) =$ `gradient_mu`.
  `= 1` The weights are adapted at each iteration using an approximation of the eigenvalues of the Hessian matrix based on the diagonalization of H^{11} (see the article for more details). This is an empiral recipe and we do not guarantee it will work for every calculation.
  `= 2` same as above but using the diagonalization of the single-particle hamiltonian h instead. This method seems more stable for odd-mass nuclei. This is an empiral recipe and we do not guarantee it will work for every calculation.
- `gradient_eta`: value of $\eta(i)$ if the weights are fixed or if the intermediate diagonalizations performed for the recipes `gradient_type = 1 or 2` fail.
- `gradient_mu`: same for $\mu(i)$.
- `gradient_eps`: convergence criterion for the gradient. The calculation will stop at iteration i if $||G(i)||_F/\eta(i) \leq$ `gradient_eps`.

Example

```
Iterative Procedure
-------------------
Maximum no.  of iterations    250
Step intermediate wf writing  0
More intermediate printing    1
Type of gradient              2
Parameter eta for gradient    1.000E-02
Parameter mu for gradient     0.500E-03
Tolerance for gradient        1.000E-04
```

1.5 Section about the constraints

Description

Line	Format	Data
35	1a	input_names(35)
36	1a	input_names(36)
37	1a30, 1i1	input_names(37), enforce_NZ
38	1a30, 1i1	input_names(38), opt_betalm
39	1a30, 1i1	input_names(39), pairs_scheme
40	1a30, 1es10.3	input_names(40), constraint_eps
41	1a30, 1i1, 1x, 1f8.3	input_names(41), constraint_switch(3), 1x, constraint_read(3)
42	1a30, 1i1, 1x, 1f8.3	input_names(42), constraint_switch(4), 1x, constraint_read(4)
⋮	⋮	⋮
63	1a30, 1i1, 1x, 1f8.3	input_names(63), constraint_switch(25), 1x, constraint_read(25)
64	1a30, 1i1, 1x, 1f8.3	input_names(64), constraint_switch(26), 1x, constraint_read(26)

where

- `enforce_NZ`: option to enforce the constraint on the average particle numbers even when doing VAPNP calculations. This may be useful to stabilize the VAPNP convergence when using a few discretization points for the projection and large steps for the gradient.
  `= 0` no constraint on the average particle numbers is enforced when doing VAPNP.
  `= 1` the code enforces a constraint on the average particle numbers of the underlying quasi-particle states.

- `opt_betalm`: option to select the kind of deformation constraints used.
  `= 0` the constraints are applied to the expectation values of $\tilde{Q}_{lm}$ directly.
  `= 1` the constraints are applied to the dimensionless parameters $\tilde{\beta}_{lm}$ (see article).
  `= 2` same as `opt_betalm = 1` except for $lm = 20$ and $lm = 22$ that are defined as the triaxial parameters β and γ, respectively (see article).

- `pairs_scheme`: option to select the kind of pair constraints used.
  `= 0` the pair operator is defined using the "agnostic" scheme (see article).
  `= 1` the pair operator is defined using the seniority scheme (see article).

- `constraint_eps`: convergence criterion during the iterative adjustement of the constraints performed at each iteration. The adjustment will stop after 100 iterations even if the criterion is not met.

- `constraint_switch(j)`: option to switch on/off the constraint `j`.
  `= 0` the constraint `j` is switched off.
  `= 1` the constraint `j` is switched on.

- `constraint_read(j)`: the expectation value targeted for the constraint `j`. Note that if `constraint_switch(j = 0`, this value is not used in the code.

The physical correspondance for the indices `j` is the following:

j	Physical quantity	
1	Z	(read as `valence_Z`)
2	N	(read as `valence_N`)
3	$\tilde{Q}_{10}$ or $\tilde{\beta}_{10}$	
4	$\tilde{Q}_{11}$ or $\tilde{\beta}_{11}$	
5	$\tilde{Q}_{20}$ or $\tilde{\beta}_{20}$ or β	
6	$\tilde{Q}_{21}$ or $\tilde{\beta}_{21}$	
7	$\tilde{Q}_{22}$ or $\tilde{\beta}_{22}$ or γ	
8	$\tilde{Q}_{30}$ or $\tilde{\beta}_{30}$	
9	$\tilde{Q}_{31}$ or $\tilde{\beta}_{31}$	
10	$\tilde{Q}_{32}$ or $\tilde{\beta}_{32}$	
11	$\tilde{Q}_{33}$ or $\tilde{\beta}_{33}$	
12	$\tilde{Q}_{40}$ or $\tilde{\beta}_{40}$	
13	$\tilde{Q}_{41}$ or $\tilde{\beta}_{41}$	
14	$\tilde{Q}_{42}$ or $\tilde{\beta}_{42}$	
15	$\tilde{Q}_{43}$ or $\tilde{\beta}_{43}$	
16	$\tilde{Q}_{44}$ or $\tilde{\beta}_{44}$	
17	J_x	
18	J_y	(not active)
19	J_z	
20	$\left[\tilde{P}^{\dagger}_{\mathrm{x}}\right]^{J=1,\,T=0}_{M_J=0,\,M_T=0}$	with x = agn or sen
21	$\left[\tilde{P}^{\dagger}_{\mathrm{x}}\right]^{J=1,\,T=0}_{M_J=-1,\,M_T=0}$	with x = agn or sen
22	$\left[\tilde{P}^{\dagger}_{\mathrm{x}}\right]^{J=1,\,T=0}_{M_J=+1,\,M_T=0}$	with x = agn or sen
23	$\left[\tilde{P}^{\dagger}_{\mathrm{x}}\right]^{J=0,\,T=1}_{M_J=0,\,M_T=0}$	with x = agn or sen
24	$\left[\tilde{P}^{\dagger}_{\mathrm{x}}\right]^{J=0,\,T=1}_{M_J=0,\,M_T=-1}$	with x = agn or sen
25	$\left[\tilde{P}^{\dagger}_{\mathrm{x}}\right]^{J=0,\,T=1}_{M_J=0,\,M_T=+1}$	with x = agn or sen
26	Δ (HFB field)	

Example

```
Constraints
-----------
Force constraint N/Z        1
Constraint beta_lm          2
Pair coupling scheme        1
Tolerence for constraints   1.000E-06
Constraint multipole Q10    0 0.000
Constraint multipole Q11    0 0.000
Constraint multipole Q20    1 0.100
Constraint multipole Q21    1 0.000
Constraint multipole Q22    1 10.00
Constraint multipole Q30    0 0.400
Constraint multipole Q31    0 0.100
Constraint multipole Q32    0 0.100
Constraint multipole Q33    0 0.100
Constraint multipole Q40    0 0.000
Constraint multipole Q41    0 0.000
Constraint multipole Q42    0 0.000
Constraint multipole Q43    0 0.000
Constraint multipole Q44    0 0.000
Constraint ang. mom. Jx     0 1.000
Constraint ang. mom. Jy     0 1.000
Constraint ang. mom. Jz     0 1.000
Constraint pair P_T00_J10   0 1.000
Constraint pair P_T00_J1m1  0 2.000
Constraint pair P_T00_J1p1  0 2.000
Constraint pair P_T10_J00   0 1.000
Constraint pair P_T1m1_J00  0 1.000
Constraint pair P_T1p1_J00  0 1.000
Constraint field Delta      0 0.050
```