BoccaDorata package for Self-consistent Green's function calculations — User manual

Carlo Barbieri

Department of Physics, University of Surrey, Guildford GU2 7XH, UK

July 2015

Contents

Forewords

The *BcDor* [1] code is built upon a C++ class library that I have developed over the past decade and that is meant for the computation of many-body Green's functions (a.k.a. propagators) in finite systems. This is written in J-coupled formalism and it is therefore mostly suitable for the *ab-initio* computation of finite nuclei in the medium mass range.

The public version of *BcDor* contains all the basic components of this library and allow for calculation of closed-shell nuclei up to *second order* in the self-energy expansion and up to the *coupled cluster with doubles* approximation. This will allow for simple computations of binding energies, of the nuclear self-energy (which provides an optical potential) and of the spectral function.

The main code is invoked from the terminal, with arguments that allow to input most parameters and to control the execution directly from the command line. It also also provides some interfaces to read file formats normally used within the nuclear theory community. This should simplify its direct use for non experts. However, it must be borne in mind that *BcDor* is not meant to be a black box tool. The user will need at least a basic understanding of Green's function and many-body theory in order to properly interpret the outputs and control the validity of the results obtained. All together, the *BcDor* can be extremely useful if used properly but the principle of *junk-in–junk-out* rules here—as always.

The public version of *BcDor* is freely available at the following weblink: `http://personal.ph.surrey.ac.uk/~cb0023/bcdor/` and this manual describes how to use it. The first chapter contains a detailed tutorial that shows how to run the code from command line and guides the user (almost hand by hand) through the procedure for performing a full calculation. This covers all the principal possibilities. Refer to the command '`BcDor -h`' to have a complete list of all possible options. The file formats used by *BcDor* for various objects (model space, propagators, etc...) are described in de-

[1]The full name *BoccaDorata* is taken from a character of comic book creator Hugo Pratt.

tails in Chapter 2. This explains how the required input data files are to be prepared.

To use the library at a deeper level, one will need some knowledge of C++ and to first learn the functioning of the three main classes that form *BcDor* : these are named `ModSpace_t`, `SpPtop_t` and `VppInt_t` in the source files. Besides its use in simple calculations and research applications, *BcDor* is an example of an advanced code for state-of-the-art *ab-initio* nuclear theory (except, perhaps, for being too short of comments right now...) and it could be a good starting point for students who want to enter the field and wish to pursue more sophisticated calculations. I would be delighted if this software could help fostering some new major advance in theoretical many-body methods.

Happy Computing!

C. Barbieri, Surrey, July 2015.

Chapter 1

Getting started

1.1 Installation

The source code is found in the folder '`source`'. Enter this directory, modify the Makefile appropriately and type '`make`'. Put the executable '`BcDor`' in your search `$PATH`. A C++ compiler and a BLAS/LAPACK library are needed.

Once the code is compiled type:

`BcDor -v`   (for the version and compilation time)

and

`BcDor -h`   (for a quick help).

1.2 Documentation and Examples

A copy of this manual is found in the '`docs`' folder.

Auxiliary files for center of mass (COM) corrections and the Coulomb interaction in a harmonic oscillator basis are in the folder '`datafiles`'. This includes also a few examples of two-nucleon interaction files that can used for experimenting. All these file formats are described in Chapter 2.

The folder '`examples`' contains the output propagators and other files that are calculated following the tutorial in the next section.

1.3 Tutorial

The results of the following calculations are in the `examples` folder, while input files of two-body matrix elements are in `datafiles`. One can run

```
> BcDor -h
```

to see all possible command line options.

BcDor assumes that the kinetic energy of the center of mass motion is to be subtracted (i.e., `Trel=3` is set by default). Unless one turns off this feature, the code will look for the relevant input file during execution. So, let's first copy this and a file with a two-body nuclear force to the folder we are working in:

```
> cp -i ../datafiles/Vpp_cdbonn_gmtx-50.0MeV_hw14MeV_Nmax7.bcd  ./Vpp.bcd

> cp -i ../datafiles/Trel-pipj_Nmax13.bcd                ./Trelpp_1.bcd
```

(the file `Trel-pipj_Nmax13.bcd` also works for much larger model spaces, up to $N_{\max}$=13). Let's now get started.

1.3.1 Templates for model spaces and propagators

To create a file for an harmonic oscillator space truncated at $N_{\max}$=7 (8 major shells), type:

```
> BcDor   MakeMdSp
```

and enter `600` (or any very large number), `7` and `2`. The model space is displayed and a file `input_msp-wt` is created, which could be renamed as follows

```
> mv -i input_msp-wt  input_msp_N7
```

The option `MakeSpProp` generates a template single-particle propagator based on the orbits in the model space. To do this, one must mark which orbits are to be considered as hole states by putting a '`1`' in the second column of the model space file (maked in red below). In order to calculate ^{16}O we will then edit the file `input_msp_N7` as follows:

```
# Definition of the model space by listing its s.p. orbitals
#-------------------------------------------------------------

# number of (ilj pi) subshells and total number of orbitals:
      30      72

# subshell:  v_s1/2
       0       1       0       0
       4       2       2
# principal q.#,  p/h character and sp energies:
       0       1    0.000     # 0v_s1/2
       1       0    0.000     # 1v_s1/2
       2       0    0.000     # 2v_s1/2
       3       0    0.000     # 3v_s1/2

# subshell:  v_p1/2
       1       1       1       0
       4       2       2
# principal q.#,  p/h character and sp energies:
       0       1    0.000     # 0v_p1/2
       1       0    0.000     # 1v_p1/2
       2       0    0.000     # 2v_p1/2
       3       0    0.000     # 3v_p1/2

# subshell:  v_p3/2
       1       3       1       0
       4       2       2
# principal q.#,  p/h character and sp energies:
       0       1    0.000     # 0v_p3/2
       1       0    0.000     # 1v_p3/2
       2       0    0.000     # 2v_p3/2
       3       0    0.000     # 3v_p3/2
       :       :      :
       :       :      :
       :       :      :
# subshell:  p_s1/2
       0       1       0       1
       4       2       2
# principal q.#,  p/h character and sp energies:
       0       1    0.000     # 0p_s1/2
       1       0    0.000     # 1p_s1/2
       2       0    0.000     # 2p_s1/2
       3       0    0.000     # 3p_s1/2

# subshell:  p_p1/2
       1       1       1       1
       4       2       2
# principal q.#,  p/h character and sp energies:
       0       1    0.000     # 0p_p1/2
       1       0    0.000     # 1p_p1/2
       2       0    0.000     # 2p_p1/2
       3       0    0.000     # 3p_p1/2

# subshell:  p_p3/2
       1       3       1       1
       4       2       2
# principal q.#,  p/h character and sp energies:
```

```
0       1    0.000     # 0p_p3/2
1       0    0.000     # 1p_p3/2
2       0    0.000     # 2p_p3/2
3       0    0.000     # 3p_p3/2
:       :      :
:       :      :
```

The following will generate a template propagator file named `sp_prop-wt`, which we rename to indicate that it is a unperturbed propagator for the ^{16}O nucleus,

```
> BcDor MdSp=input_msp_N7  MakeSpProp

> mv  -i  sp_prop-wt sp_prop_O16_unpert
```

1.3.2 Hartree-Fock propagator

We are now in the position to solve the Hartree-Fock (HF) equations. To do this, we must tell the mass number, the number of protons and neutrons, and the harmonic oscillator frequency ($\hbar\Omega$=14 MeV for the two-body interactions file we choose). The following command will perform one iteration of the HF equations (to be entered all on one line):

```
> BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                          SpProp=sp_prop_O16_unpert  HF
```

The final energy, the number of nucleons and the list of unoccupied/occupied orbits is listed at the end of the screen output. The input values of `A`, `N` and `Z` are required but they are used only to determine the correct coefficients for the center of mass corrections (in this case only `A` is relevant). The propagator resulting from this calculation is written in an output file named `sp_prop-SC-A=16_itr1-wt`. Beware that *BcDor* sometimes interprets certain solutions as occupied states when they are not or vice versa. In these cases one can fix this by forcing the Fermi level of a given partial wave `i` with the option `SetEf=i,Ef`. This should not be needed for the present example.

The option `HF` executes only one iteration. One can state `HF=i` to iterate `i` times or `HF=-1` to iterate until convergence. Let's do the latter:

```
> BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                     SpProp=sp_prop_O16_unpert  HF=-1
```

At the end of the execution, *BcDor* generates a propagator file named with the result of the last iteration. This has always the same name, with a '`-999`' in it, independently of how many iterations have been performed. Since this is our convergent HF solution, let's save it with a proper name:

```
>  mv -i sp_prop-SC-A\=16_itr-999-wt sp_prop_O16_HF
```

Finally, the `Koltun` option can be used to calculate the Koltun sum rule for the total energy,

```
> BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                             SpProp=sp_prop_O16_HF  Koltun
```

The Koltun sum rule is exact for two-body interactions and for the exact one-body propagator. When applied to a Hartree-Fock state, it is related to the Koopmans theorem and it simplifies to calculate the HF energy as a sum of the single particle energies for the occupied orbits. Hence, the `Koltun` option should give exactly the same results of the convergent HF run above.

The expectation values of kinetic energy (with T_{COM} subtracted as we are implicitly using the `Trel=3` option), of the two-body interaction and the total energy in HF approximations are:

$$\begin{aligned} \langle \hat{T} \rangle &= \quad 336.534\,\text{MeV}, \\ \langle \hat{V} \rangle &= -398.653\,\text{MeV}, \\ E_{Koltun} = E_{HF} &= -62.1194\,\text{MeV}. \end{aligned} \tag{1.1}$$

1.3.3 Coupled cluster and perturbation theory

The HF propagator just obtained can be used to calculate the correlation energy from second order many-body perturbation theory. The option `MBPT2` works exactly as for the `Koltun` case but it also calculates the extra contributions at second order:

```
> BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                                  SpProp=sp_prop_O16_HF  MBPT2
```

In the output from this calculations, the value marked as '`E2 = ...`' is the MBPT(2) contribution to the correlation energy.

In the present version of the code, the coupled cluster method has been implemented just for the simplest approximation of only doubles (this may change in the future). Furthermore, beware that *BcDor* assumes an HF reference state: giving a different input than HF would neglect the non-diagonal f_{ab} and f_{ij} terms in the CCD equations.

Since we do have an HF propagator, we can do the correct CCD calculations by simply typing:

```
> BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                                    SpProp=sp_prop_O16_HF  CCD
```

BcDor will first build the unperturbed T^2 amplitudes and then start iterations. As usual, the value before the first iterations is the MBPT(2) approximation to the correlation energy. The converged energy is instead the correlation energy in the CCD approximation.

The present coupled cluster option also allows for linear mixing and for switching on the two-body interaction in an adiabatic way (which may help only in particular cases, when the method is on the brink of diverging). However, no other convergence accelerators are implemented at the moment.

To summarise the correlations energies in MBPT(2) and CCD are:

$$E^{(2)}_{corr} = -59.228\,\text{MeV}, \quad \text{and} \quad E^{CCD}_{corr} = -59.005\,\text{MeV}. \tag{1.2}$$

1.3.4 One-body propagator

The option `2nd` will solve the Dyson equation with a self-energy calculated up to second order in perturbation theory. This results in an all-orders, nonperturbative, resummation of the nuclear self-energy. In the following example, both the energy-independent part, $\Sigma^{(\infty)}_{\alpha\beta}$, and the second order diagram, $\Sigma^{(2)}_{\alpha\beta}(\omega)$, are calculated based on the input HF propagator. They are generated for each partial wave present in the model space and then diagonalised. Since this procedure elaborates much more information than just the correlation energy, it will take a few minutes to go through (again, all entered on one line):

```
> BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                         SpProp=sp_prop_O16_HF  2nd
```

As each partial wave is diagonalised, *BcDor* will output to screen the Dyson orbits surrounding the Fermi surface, together with partial sums of the Koltun sum rule. Note that $\Sigma^{(\infty)}_{\alpha\beta}$ is calculated as the 'correlated HF' (cHF) diagram (or tadpole diagram) and the input propagator is used to do this at the first iteration. In the above example, this simplifies to the normal HF potential since we have used the HF results as an input. Thus, the second order contribution to the energy independent self-energy is neglected and this calculation is strictly speaking *not* a compete ADC(2) truncation. The next subsection describes how to calculate this missing term (and much more than that) self-consistently.

The correlated one body Green's function obtained by the one iteration performed above is written to a file called '`sp_prop-SC-A=16_itr1-wt`'. We can save this with a more meaningful name and recalculate the Koltun sum rule at any later time to obtain the total energy,

```
> mv -i sp_prop-SC-A=16_itr1-wt sp_prop_O16_2nd
```

```
> BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                      SpProp=sp_prop_O16_2nd  Koltun
```

The resulting value for the ground state energy is $E^{2nd}_{g.s.}$=-134.846 MeV.

The file '`sp_prop_O16_2nd`' now contains the fully dressed propagator, from which the full one-particle and one-hole spectral functions can be extracted and plotted. The energy differences for the addition and removal of a nucleon are clearly indicated in the file together with the corresponding spectroscopic factors (not necessarily converged at second order and in the small model space used here). Refer to Section 2.2 for details about the format of this file. Command options for facilitating the graphical plots of propagators are not available at the moment but may be included in future distributions of the software.

The self-energy used for the above calculations can also be calculated separately and saved to file using the '`MakeSelfEn`' option. *BcDor* saves this information in separate files (one for each partial wave) inside a folder named '`bcdwk`' by default (see the '`-h`' option to change this name). This is done by typing,

```
> mkdir bcdwk

> BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                  SpProp=sp_prop_O16_HF  MakeSelfEn  2nd
```

where '`MakeSelfEn`' tells *BcDor* to only calculate the self-energy and save it without diagonalising the Dyson equation. The '`2nd`' is needed to tell what approximation for $\tilde{\Sigma}_{\alpha\beta}(\omega)$ is to be used.

The self-energy is now stored in `bcdwk` as binary files:

```
 > ls -l bcdwk/
total 7392
-rw-r--r--  1 user  group   90244 26 Jul 14:24 SelfEn_Dyn_J2pi=1+_dt=0.bin
-rw-r--r--  1 user  group   90244 26 Jul 14:24 SelfEn_Dyn_J2pi=1+_dt=1.bin
-rw-r--r--  1 user  group   90868 26 Jul 14:24 SelfEn_Dyn_J2pi=1-_dt=0.bin
-rw-r--r--  1 user  group   90868 26 Jul 14:24 SelfEn_Dyn_J2pi=1-_dt=1.bin
-rw-r--r--  1 user  group   93268 26 Jul 14:24 SelfEn_Dyn_J2pi=11+_dt=0.bin
-rw-r--r--  1 user  group   93268 26 Jul 14:24 SelfEn_Dyn_J2pi=11+_dt=1.bin
    :           :                     :            :
    :           :                     :            :
```

The is an option, `PlotSelfEn`, that can be used to cast these files in human readable text format.

The self-energy just calculated can be used to solve the Dyson equation for the one-body propagator. In the following case, the option '`ExtSE`' tells *BcDor* to perform one single iteration of the Dyson equation still using the propagator '`sp_prop_O16_HF`' to calculate $\Sigma^{(\infty)}_{\alpha\beta}$ but taking the energy dependent self-energy form the `bcdwk` folder:

```
>BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                            SpProp=sp_prop_O16_HF  ExtSE
```

Note that this is different from the above calculation since the input propagator is used *only* to initiate the calculation of $\Sigma^{(\infty)}_{\alpha\beta}$. However, since the the self-energy present in '`bcdwk`' was also calculated at second order from the HF reference state, this last example will give exactly the same output of the previous calculation.

1.3.5 Lanczos acceleration and self-consistency

For large model space, the diacgonalization of the Dyson equation slows down due to the large number of $2p1h$ and $2h1p$ configurations contributing to the poles of the self-energy. This can be accelerated using a Lanczos algorithm to replace the self-energy with an effective one having a much smaller number of poles. The reader is refereed to Phys. Rev. C**89**, 024323 (2014) for all technical details of this procedure.

We first delete the files already contained in '`bcdwk`'. The procedure is to generate a file with pivot vectors for the Lanczos iterations, to build the new reduced self-energy using Lanczos, and then to diagonalise the Dyson equation as done previously but with the new self-energy. A number of 50 iteration per pivot will be sufficient to obtain an accurate result for this example. In sequence:

```
> rm -f  bcdwk/Sel*

>BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                           SpProp=sp_prop_O16_HF   MakePivots=50,Pivts.dat

>BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                              SpProp=sp_prop_O16_HF  MakeSelfEn  2nd
                                              LanDysPiv=Pivts.dat  Lanczos

>BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                              SpProp=sp_prop_O16_HF  ExtSE
```

As usual, one can rename the calculated propagator and perform the Koltun sum rule:

```
mv -i  sp_prop-SC-A\=16_itr1-wt sp_prop_O16_2nd_Lanc50

>BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                      SpProp=sp_prop_O16_2nd_Lanc50 Koltun
```

Comparing to the results in the previous section one can see that there is no loss of accuracy in calculated energies:

Full self-energy:	Lanczos reduced self-energy:
$\langle\hat{T}\rangle = \quad 366.529\,\text{MeV}$	$\langle\hat{T}\rangle = \quad 366.528\,\text{MeV}$
$\langle\hat{V}\rangle = -501.375\,\text{MeV}$	$\langle\hat{V}\rangle = -501.374\,\text{MeV}$
$E^{2nd}_{Kolt} = -134.846\,\text{MeV}$	$E^{2nd}_{Kolt} = -134.847\,\text{MeV}$

At this point it is instructive to inspect the two propagator files '`sp_prop_O16_2nd`' and '`sp_prop_O16_2nd_Lanc50`' and to compare how they differ in size and how much the Dyoson orbitals have changed close to the Fermi energy.

Now that the Dyson diagonalization has been speeded up, it can be iterated to reach a self-consistent solution for $\Sigma^{(\infty)}_{\alpha\beta}$. This approximation is referred to as 'sc0' and resums all order contributions to the energy independent self-energy, including second-, third-order, and beyond [see Phys. Rev. C**89**, 024323 (2014) for a full discussion]. As for the `HF` option, '`ExtSE=-1`' iterates the Dyson equations until convergence:

```
>BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                          SpProp=sp_prop_O16_HF  ExtSE=-1

> mv -i   sp_prop-SC-A\=16_itr-999-wt sp_prop_O16_2nd_sc0

> BcDor  MdSp=input_msp_N7  A=16  Z=8  N=8  hwHO=14.0  Vpp=Vpp.bcd
                                     SpProp=sp_prop_O16_2nd_sc0    Koltun
```

where we obtain $E^{ADC(2)}_{g.s.}$=-122.072 MeV for the total energy in the ADC(2)-sc0 approximation.

The following table summarises all the results obtained above:

```
HF:            E_0    =  -62.119 MeV

MBPT2:         E_2    =  -59.228 MeV        E_tot  =  -121.347 MeV

CCD:           E_corr =  -59.005 MeV        E_tot  =  -121.124 MeV

GF(2nd ord - 1 itr.)                        E_kolt =  -134.846 MeV

ADC(2)-sc0                                  E_kolt =  -122.072 MeV
```

Chapter 2

Types of data files

Microscopic many-body calculation for systems such as atomic nuclei require three input ingredients: the Hilbert space (here defined through a single particle model space), the Hamiltonian (here given by matrix elements of a two-body interaction) and a reference state upon which the complete wave function or propagator are constructed.

Correspondingly, *BcDor* reads three main types of data files for the *model space*, the *two-body interaction* and the *one-body propagator* $g_{\alpha\beta}(\omega)$. Propagator files are used both to define the input (unperturbed) reference states and to store the actual calculated propagator (hence, the spectral function).

2.1 Model Space

The file that describes the model space is passed to the code through the command line option `MdSp=filename`. The *BcDor* code is written to work in in J-coupled formalism. Thus, it expects single particle orbits in a spherical basis where orbital momentum ℓ and spin s are coupled to a total angular momentum j. In the vast majority of applications this will be a spherical harmonic oscillator basis. However, the the code is *not* limited to this space and can work with any discretised spherical basis.

Each single particle orbit is $\{\alpha\}$ is identified by the pincipal quantum number n_α, the parity π_α (=0 for positive and =1 for negative parity), the total angular momentum j_α and the charge e_α (-1, 0, +1 for electrons, neutrons and protons). The latter three quantum numbers represent symmetries of the total Hamiltonian and therefore are conserved for quasiparticle and quasihole excitations (a J=0 ground state is always assumed here). The model space is therefore grouped by *partial waves*, or '*subshells*', of given parity, angular momentum, and charge: $a \equiv (\pi_\alpha,\ j_\alpha,\ e_\alpha)$. The orbits belonging to each partial wave are distinguished by their n_α and the full set of quantum numbers is $\alpha \equiv(n_\alpha,a)=(n_\alpha,\ \pi_\alpha,\ j_\alpha,\ e_\alpha)$. The magnetic quantum number m_{j_α} is implicitly assumed and does not need to be specified to define the model space.

The following example demonstrates the structure of the input file required to describe the model space:

```
# Definition of the model space by listing its s.p. orbitals
#--------------------------------------------------------------

# number of (ilj\pi) subshells and total number of orbitals:
      30       72

# subshell:  v_s1/2
       0       1       0       0
       4       2       2
# principal q.#,  p/h character and sp energies:
       0       1    0.000     # 0v_s1/2
       1       0    0.000     # 1v_s1/2
       2       0    0.000     # 2v_s1/2
       3       0    0.000     # 3v_s1/2

# subshell:  v_p1/2
       1       1       1       0
       4       2       2
# principal q.#,  p/h character and sp energies:
       0       1    0.000     # 0v_p1/2
       1       0    0.000     # 1v_p1/2
       2       0    0.000     # 2v_p1/2
       3       0    0.000     # 3v_p1/2

# subshell:  v_p3/2
       1       3       1       0
       4       2       2
# principal q.#,  p/h character and sp energies:
       0       1    0.000     # 0v_p3/2
       1       0    0.000     # 1v_p3/2
       2       0    0.000     # 2v_p3/2
       3       0    0.000     # 3v_p3/2
       :       :      :
       :       :      :
       :       :      :
```

In details:

- The first 4 lines are comments and are not read by the code.

- The s 5-th line contains the numbers of different partial waves ($j\pi e$) that form the model space and the total number single particle orbits ($nj\pi e$), not counting the m_j degeneracy.

- After a blank line, there must be a series of blocks describing each separate partial wave. These block are separated by a blank line.

- For each partial wave block:

  - The first line is a comments, e.g. giving a human readable description of the partial wave.
  - The second line contains, in order, the quantum numbers ℓ_α, $2j_\alpha$, π_α and e_α. Note that this code is for many-fermion systems, thus the *integer* value of $2j$ must be given. The quantum number ℓ must be specified but it is redundant for a set of spin-$\frac{1}{2}$ particles since it is already constrained by j and π. For the *BcDor* code to work correctly, the π quantum number is the one that matters.
  - The third line is expected to contain three integer numbers. The first is the number of orbits (differing by their principal quantum number n) that belong to this partial wave. The remaining two numbers are redundant and not used but must be specified.
  - The fourth line is a comment.
  - Each of the following lines describes a different orbits and contains three numbers: one *integer* giving the principal quantum number n_α, a second *integer* value and a *floating point* value. The last two values are normally not used but become handy in some situations. For example, the second integer value is used to mark which orbits should be interpreted as particles or holes when generating a template propagator (see option `MakeSpProp` in the tutorial at Sec. 1.3.1).

Note that the order of the partial wave blocks within the file is arbitrary. Likewise, the single particle orbits within each block do not need to be ordered according to the principal quantum number. The data files for the interaction (see section 2.3) must contain the physical value of quantum number for each matrix element. *BcDor* reads this information and links it automatically to the correct orbit of the model space. *However*, the one-body propagator files must contain partial wave blocks in the same order of the model space file (section 2.2). And, likewise, the spectroscopic amplitudes corresponding to different values of n_α must be listed with the same order in both files. With this caveat, *BcDor* accepts input data with arbitrary ordering:

```
       :       :      :
       :       :      :

# subshell:  p_f7/2
       3       7       1       1
       6       2       1
# principal q.#,  p/h character and sp energies:
       0       1    0.000     # 0p_f7/2
       5       0    0.000     # 5p_f7/2
       2       0    0.000     # 2p_f7/2
       4       0    0.000     # 4p_f7/2
       3       0    0.000     # 3p_f7/2
       1       0    0.000     # 1p_f7/2

# subshell:  v_d3/2
       2       3       0       0
       3       2       1
# principal q.#,  p/h character and sp energies:
       2       0    0.000     # 2v_d3/2
       1       1    0.000     # 1v_d3/2
       0       1    0.000     # 0v_d3/2

       :       :      :
       :       :      :
```

2.2 One-body propagator

The one-body propagator (and hence the spectral function) is stored in terms of its poles and spectroscopic amplitudes. Since *BcDor* works in a discretised model space, its Lehman representation takes the form of a discrete sum of poles:

$$g_{\alpha\beta}(\omega) = \sum_n \frac{(\mathcal{X}^n_\alpha)^* \mathcal{X}^n_\beta}{\omega - \varepsilon^+_n + i\eta} + \sum_k \frac{\mathcal{Y}^k_\alpha (\mathcal{Y}^k_\beta)^*}{\omega - \varepsilon^-_k - i\eta} \tag{2.1}$$

where we used the following notation for the quasiparticle fragments

$$\begin{cases} \mathcal{X}^n_\alpha \equiv \langle \Psi^{A+1}_n | a^\dagger_\alpha | \Psi^A_0 \rangle \\ \varepsilon^+_n \equiv E^{A+1}_n - E^A_0 \\ E^{A+1}_n | \Psi^{A+1}_n \rangle = H | \Psi^{A+1}_n \rangle \end{cases} \tag{2.2}$$

and for the quasihole fragments

$$\begin{cases} \mathcal{Y}_\alpha^k & \equiv & \langle \Psi_k^{A-1} | a_\alpha | \Psi_0^A \rangle \\ \\ \varepsilon_k^- & \equiv & E_0^A - E_k^{A-1} \\ \\ E_k^{A-1} | \Psi_k^{A-1} \rangle = H | \Psi_k^{A-1} \rangle \end{cases} \tag{2.3}$$

In these notations greek indices $\{\alpha\}$ label the set of single particle orbits that define the model space, while the n (k) indices label the exact eigenstates of the Hamiltonian H for $A+1$ $(A-1)$ particle numbers.

Data files for single particle propagators are passed to *BcDor* with the command option `SpProp=filename` and must take the following form:

```
#  Quasi- particle and hole fragments of the sp propagator
#-----------------------------------------------------------

# number of (ilj\pi) subshells, max n. of radial orbitals:
#      30       4       0

# Total numbers of qp and qh stored here:
#      66       6

# Subshell:
# v_s1/2
#       3  # -> tot n. of quasiparticles
       48.340786      100.000       -1.082472e-02      -1.143617e-01       4.154943e-01       9.023130e-01
       17.932187      100.000       -1.192554e-01       2.059822e-01       8.913622e-01      -3.857757e-01
       -0.108800      100.000        2.451556e-01      -9.356775e-01       1.659063e-01      -1.920454e-01
#       1  # -> tot n. of quasiholes
      -57.898314      100.000        9.620601e-01       2.626792e-01       7.288995e-02       1.127006e-02

# Subshell:
# v_p1/2
#       3  # -> tot n. of quasiparticles
       64.569272      100.000       -7.720331e-02      -1.107207e-02       5.302053e-01       8.442745e-01
       29.605829      100.000       -1.570891e-01       5.226215e-01       7.062331e-01      -4.510261e-01
        8.387328      100.000        1.052361e-01      -8.348091e-01       4.570321e-01      -2.883414e-01
#       1  # -> tot n. of quasiholes
      -16.365735      100.000        9.789219e-01       1.727364e-01       1.060134e-01       2.520469e-02

# Subshell:
# v_p3/2
#       3  # -> tot n. of quasiparticles
       60.594865      100.000       -6.423154e-02      -5.378335e-02       4.982688e-01       8.629657e-01
       27.308224      100.000       -1.948788e-01       4.504723e-01       7.603290e-01      -4.254372e-01
        7.320176      100.000        1.906771e-01      -8.532102e-01       4.025175e-01      -2.713931e-01
#       1  # -> tot n. of quasiholes
      -26.133366      100.000        9.599681e-01       2.573217e-01       1.077387e-01       2.528143e-02

# Subshell:
# v_d3/2
#       3  # -> tot n. of quasiparticles
       54.304538      100.000       -3.331303e-02       4.839809e-01       8.744442e-01
```

```
        22.642110      100.000         3.797932e-01       8.154270e-01      -4.368478e-01
         6.986860      100.000         9.244714e-01      -3.175552e-01       2.109770e-01
#        0  # -> tot n. of quasiholes

         :                :                :
         :                :                :
```

The propagator is also diagonal in the $(j\pi e)$ quantum numbers and therefore splits in separate partial wave blocks. In details:

- The first 4 lines are of the file are comments and are not read.
- Line 5 reports the total number of partial wave blocks to be read and the maximum number of different orbits for each block (that is, the dimension of the largest block in the model space). These numbers *must* be the same as the corresponding values for the associated model space. A trailing '`#`' character is expected.
  The third number is an advanced feature to provide extra information and ot is normally not needed. This number should not be given at all or set to zero ('`0`'). If *BcDor* cannot read it it automatically sets it to zero (and one doesn't need to worry any further).
- Lines 6-7 are comments.
- Line 8 gives the total number of quasiparticle and quasihole poles contained in the file. *BcDor* checks that these are consistent with the content of the file and gives a warning if they are not.
- After a blank line, there must be a series of blocks describing each separate partial wave. These block are separated by a blank line and must come *exactly* in the same ordered as they are listed in the model space file (see section 2.1).
- For each partial wave block:
  - The first two lines are comments, e.g. giving a human readable description of the partial wave.
  - The 3rd line gives the numbers of quasiparticle poles for this given partial wave (trailing '`#`' expected).
  - The following lines list the quasiparticle fragments (one on each line), listing the fragment's energy, the spectroscopic factor (in %) and the spectroscopic amplitudes:

$$\varepsilon_n^+ \qquad Z_n^+ \qquad \mathcal{X}_\alpha^n \quad \mathcal{X}_{\alpha'}^n \quad \mathcal{X}_{\alpha''}^n \quad \mathcal{X}_{\alpha'''}^n \quad \ldots$$

The amplitudes $\mathcal{X}^n_\alpha$ for the same fragment depend on the single particle orbit (specifically, its principal quantum number n_α) and *must* follow the same order given in the model space description (see section 2.1).

- The next line gives the numbers of quasihole poles for this given partial wave (trailing '#' expected).
- The following lines list the quasiparticle fragments (one on each line), listing the fragment's energy, the spectroscopic factor (in %) and the spectroscopic amplitudes:

$$\varepsilon^-_k \qquad Z^-_k \qquad \mathcal{Y}^k_\alpha \quad \mathcal{Y}^k_{\alpha'} \quad \mathcal{Y}^k_{\alpha''} \quad \mathcal{Y}^k_{\alpha'''} \quad \dots$$

  Likewise, the amplitudes $\mathcal{Y}^n_\alpha$ for the same fragment depend on the single particle orbit and must come exactly in the same order as in the associated model space.

Be warned that, within each block, it is *very important* that the number of quasiparticle and quasi hole poles stated at the beginning of the list correspond to the exact number of poles listed in the following lines. If any of these values is wrong, it will result in wrong execution of the program and most likely in a segmentation fault (a beginner user may want to experiment with a wrong input once or twice to learn recognising this situation, should it occur in the future...).

Spectroscopic factors are obtained by integrating the squared spectroscopic amplitudes over the single particle basis and are intended here as percentage of the independent particle model (or mean-field) value:

$$Z^+_n[\%] = 100 \times \sum_\alpha |\mathcal{X}^n_\alpha|^2 \;, \qquad Z^-_k[\%] = 100 \times \sum_\alpha |\mathcal{Y}^k_\alpha|^2 \;. \tag{2.4}$$

Note that the values in this file are mostly for output information, while the spectroscopic amplitudes are those actually used to performing calculations. The set of values $\{Z^+,\mathcal{X}\}$ and $\{Z^-,\mathcal{Y}\}$ given as input should in principle be consistent but *BcDor* does not normally check this: it simply reads in all the value values and then carries on without controlling nor recalculating the Z^+/Z^- values. If the latter are given wrongly, this has not effects in most of the cases but there is no guarantee of this.

2.3 Interactions data files

The matrix elements for a two-body interaction are passed through the command option `Vpp=filename`. *BcDor* has two native file formats for the two-body interaction that allow to give the matrix elements as a (human readable) text file or through a binary file (which is less versatile but more compact and allows for faster disk I/O). The code distinguishes between different types of formats based on the file extension: `.bcd` or `.dat` for the text format and `.bin` is for the binary format. The command option `ConvVpp=newfile.ext` allows to convert files from one format to another.

In addition to the native formats, *BcDor* can accept two-body matrix elements files in other formats. In particular, matrix elements generated by the CENS codes, written by M. Hjorth-Jensen at Oslo, can be read through files ending with the extension `.mhj`.

Interaction files in the format used by the V_{UCOM} collaboration in Darmstadt are not supported by *BcDor* . However there exist a separate program that performs the conversion into the `.bcd` format (contact me if interested).

2.3.1 Native *BcDor* interaction files (`.bcd` and `.bin`)

Text files with either `.bcd` or `.dat` extension must contain two-body matrix elements in *properly normalised* J-coupled and proton-neutron form. For each matrix element the *values* of each relevant quantum number (and not their order in the model space file) have to be used so that the same interaction can be read completely independently of the details of the model space file being inputted (see section 2.1).

Precisely, the matrix elements must be coupled to total angular momentum J and normalised as follows:

$$\bar{v}^J_{\alpha\beta,\gamma\delta} = \frac{1}{\sqrt{1+\delta_{\alpha\beta}}}\frac{1}{\sqrt{1+\delta_{\beta\delta}}}\sum_{m_\alpha m_\beta m_\gamma m_\delta}(j_\alpha\; j_\beta\; m_\alpha\; m_\beta|JM) \\ \times\langle\alpha m_\alpha\beta m_\beta|\hat{V}|\gamma m_\gamma\delta m_\delta\rangle(j_\gamma\; j_\delta\; m_\gamma\; m_\delta|JM)\;, \quad (2.5)$$

where the anti-symmetrized matrix elements are calculated for the model space orbits $\phi_{\alpha m_\alpha}(\mathbf{x}_1)$ according to [$\mathbf{x}\equiv(\vec{r},s,\tau)$ labels the spatial, spin and isospin coordinates]:

$$\langle\alpha m_\alpha\beta m_\beta|\hat{V}|\gamma m_\gamma\delta m_\delta\rangle = \int d\mathbf{x}_1\int d\mathbf{x}_2\int d\mathbf{x}_3\int d\mathbf{x}_4 \\ \phi^*_{\alpha m_\alpha}(\mathbf{x}_1)\phi^*_{\beta m_\beta}(\mathbf{x}_2)\hat{V}[\phi_{\gamma m_\gamma}(\mathbf{x}_3)\phi_{\delta m_\delta}(\mathbf{x}_4)-\phi_{\delta m_\delta}(\mathbf{x}_3)\phi_{\gamma m_\gamma}(\mathbf{x}_4)]\;. \quad (2.6)$$

When working with spherical harmonic oscillator bases, *BcDor* assumes the phase convention that $\phi_{nl}(r) > 0$ for $r \to 0$ in coordinate space (or $\tilde{\phi}_{nl}(k) \sim (-1)^n$ at small k in momentum space). This is the most widely used convention and it is needed only to calculate the matrix elements of the kinetic energy and response functions in ph-RPA calculations. In principle, one could use any convention for the two-body matrix elements but then the correct matrix elements of the kinetic energy have also to be provided by the user as an external one-body operator.

The `.bcd` file containing the two-body matrix elements must have the following format:

```
 1358286 1500000  # tot number of mtx el. stored and to be allocated; ...
0 1 0 0 0 1 0 0  0 1 0 0 0 1 0 0  0 -6.5369194750
0 1 0 0 0 1 0 0  0 1 0 0 1 1 0 0  0 -3.7886995480
0 1 0 0 0 1 0 0  0 1 0 0 2 1 0 0  0 -1.3750089480
0 1 0 0 0 1 0 0  0 1 0 0 3 1 0 0  0 -0.4418042008
   :       :        :      :      :       :
   :       :        :      :      :       :
```

The two integer numbers in the first line specify the total number of lines in the file that have to be read (one line for each matrix element) and the total number of memory locations to be allocated in RAM for the two-body interaction. Hence, it is possible to reserve more space in memory than it is actually needed to store all the matrix elements on file. This option is useful in cases when one wants to first load one interaction and then add new matrix elements corresponding to configurations that are not present in the initial file. If *BcDor* cannot read the number of slots to be allocated (e.g., the file provides only the first number) or if this is too small, it will still allocate as many slots as the matrix elements present in the file.

Each of the following lines provides a matrix elements together with the all relevant quantum numbers in the following format:

$$n_\alpha \; 2j_\alpha \; \pi_\alpha \; e_\alpha \quad n_\beta \; 2j_\beta \; \pi_\beta \; e_\beta \quad n_\gamma \; 2j_\gamma \; \pi_\gamma \; e_\gamma \quad n_\delta \; 2j_\delta \; \pi_\delta \; e_\delta \quad J \quad \bar{v}^J_{\alpha\beta,\gamma\delta}$$

Thus, the first element in the example above is between four neutrons all in the $0s_{1/2}$ state and coupled to total angular momentum J=0. The second element has two $0s_{1/2}$ neutrons on one side and one $0s_{1/2}$ plus one $1s_{1/2}$ neutron on the other side. The matrix elements do not need to be in any particular order, *BcDor* will sort them after reading the file.

The J-coupled matrix elements in Eq. (2.5) are hermitian and antisymmetric in the exchange of two nucleons [that is: $\bar{v}^J_{\alpha\beta,\gamma\delta} = (-1)^{1+j_\alpha+j_\beta-J}\bar{v}^J_{\beta\alpha,\gamma\delta} = (\bar{v}^J_{\gamma\delta,\alpha\beta})^*$], so that not all configurations are independent. Of all these configuration only one need to be included in the file. It is responsibility of the user to ensure that this is done correctly. After reading matrix elements from the `.bcd` file, *BcDor* can perform some tests to find out whether some configurations have been inputted twice or not all the possible configurations are included. If so, it gives warnings. However, keep in mind that these are quick and rough checks and they are not fail-proof.

The binary interactions files can be generated by first reading an interaction and then converting it as follows:

```
> BcDor Trel=0 MdSp=msp_file Vpp=filename_in.bcd ConvVpp=filename_out.bin
```

This makes a binary file which is an exact image of the RAM memory used to store the interactions. Therefore, it is much smaller in size and much faster to read than the corresponding text file. However, it can *only* be employed with exactly the same model space used to generate it.

2.3.2 Matrix elements for Coulomb and center of mass corrections

There are a number of auxiliary files that are used to add the Coulomb matrix elements (option '`Vc`', if the nuclear potential does *not* already contain Coulomb), to correct for the center of mass kinetic energy (options '`Trel=`', '`Trel_pipj=`' and '`Trel_2body=`') and to correct for the center of mass motion when calculating rms radii (options '`Radii=`' and '`Rrms=`'). All these files are in `.bcd` format and are expected to contain matrix elements appropriately rescaled by the harmonic oscillator length or frequency. Thus, only one set of matrix elements is needed when working with spherical harmonic oscillator bases. Different bases can of course be used but that would require calculating the relevant auxiliary matrix elements case by case.

If you are making calculations in a spherical harmonic oscillator basis, use the files given in the folder `datafiles`. That is all you need.

2.3.3 Interaction files from the CENS pakage

The *computing environment for nuclear structure* (CENS) is a set of codes for effective interactions and shell-model calculations developed at the university of Oslo, available at `http://folk.uio.no/mhjensen/cp/software.html`. This also contains software to generate matrix elements of realistic nuclear forces, both bare and renormalised (G-matrix, free space SRG, Lee-suzuki, etc...).

The CENS `vrenorm` code normally generates two files, one with a table for the model space and one with the corresponding matrix elements. The *BcDor* 's `.mhj` format must contain both of these concatenated. For example, if the model space file generated by CENS is `spdata_file.dat` and the interaction file is `vint.dat` one should create the input file as

```
> cat    spdata_file.dat  vint.dat  >  interaction_file.mhj
```

which can then be used by *BcDor* :

```
> BcDor   MdSp=msp_file   Vpp=interaction_file.mhj  ...
```

This procedure works correctly with the CENS distribution as of April 2015.

Keep in mind that the model space used in the calculation is `msp_file` and *not* `spdata_file.dat`. *BcDor* reads the latter through the `.mhj` file and the uses it only to import the matrix elements and to match them to its standard model space (`msp_file` in this example). Sometimes CENS is used to generate matrix elements of a G-matrix at different energies. In this cases *BcDor* reads all sets of matrix elements and then one wants to use the '`SelInt=i`' option (with `i=0,1,2,...`) to chose a particular starting energy. For example,

```
> BcDor Trel=0 MdSp=msp_file Vpp=Gmatrix_file.mhj SelInt=2 ConvVpp=Gmtx_Exx.bcd
```

reads the G-matrix for the 3rd starting energy stored in `Gmatrix_file.mhj` and casts it in a native `.bcd` file.

Chapter 3

Features of the unpublished Code

3.1 Further file formats

3.1.1 Three-nucleon force

Matrix elements for three-body interations (TBI) are stored in unformatted binary data files as single precision numbers and are expected to be coupled to total angular momentum and isospin (i.e. we use good isospin formalism and *not* the proton-neutron format for the TBI). The matrix elements must also be fully antisimmetrized but *not normalised.*

Thus, the full expression used for coupling the TBI elements in total angular momentum J and isospin T is

$$\begin{aligned} w^{J\,T}_{\alpha\beta J_{ab}T_{ab}\gamma\,,\,\mu\nu J_{mv}T_{mv}\lambda} = & \sum_{m_\alpha m_\beta m_\gamma M_{ab}} \; \sum_{m_\mu m_\nu m_\lambda M_{mv}} \; \sum_{\tau_\alpha \tau_\beta \tau_\gamma T^z_{ab}} \; \sum_{\tau_\mu \tau_\nu \tau_\lambda T^z_{mv}} \\ & (j_\alpha \; j_\beta \; m_\alpha \; m_\beta | J_{ab} M_{ab}) \; (J_{ab} \; j_\gamma \; M_{ab} \; m_\gamma | JM) \\ & (\frac{1}{2} \; \frac{1}{2} \; \tau_\alpha \; \tau_\beta | T_{ab} T^z_{ab}) \; (T_{ab} \; \frac{1}{2} \; T^z_{ab} \; \tau_\gamma | T T_z) \\ & \langle \alpha m_\alpha \tau_\alpha \; \beta m_\beta \tau_\beta \; \gamma m_\gamma \tau_\gamma | \hat{W} | \mu m_\mu \tau_\mu \; \nu m_\nu \tau_\nu \; \lambda m_\lambda \tau_\lambda \rangle \\ & (j_\mu \; j_\nu \; m_\mu \; m_\nu | J_{mv} M_{mv}) (J_{mv} \; j_\lambda \; M_{mv} \; m_\lambda | JM) \\ & (\frac{1}{2} \; \frac{1}{2} \; \tau_\mu \; \tau_\nu | T_{mv} T^z_{mv}) (T_{mv} \; \frac{1}{2} \; T^z_{mv} \; \tau_\lambda | T T_z) \; , \qquad (3.1) \end{aligned}$$

where the greek indices, $\alpha \equiv (n_\alpha, j_\alpha, \pi_\alpha)$, here refer to just the principal quantum number, total angular momentum and parity of the sigle particle basis (since the isospin variale are also coupled now). The matrix elements in Eq. (3.1) must be anti-symmetrized and they are calculated for the model space orbits $\phi_{\alpha m_\alpha \tau_\alpha}(\mathbf{x}_1)$ according to [$\mathbf{x} \equiv (\vec{r}, s, \tau)$ labels the spatial, spin and isospin coordinates as usual]:

$$
\begin{aligned}
&\langle \alpha m_\alpha \tau_\alpha \; \beta m_\beta \tau_\beta \; \gamma m_\gamma \tau_\gamma | \hat{W} | \mu m_\mu \tau_\mu \; \nu m_\nu \tau_\nu \; \lambda m_\lambda \tau_\lambda \rangle = \\
&\int d\mathbf{x}_1 \int d\mathbf{x}_2 \int d\mathbf{x}_3 \int d\mathbf{x}_4 \int d\mathbf{x}_5 \int d\mathbf{x}_6 \;\; \phi^*_{\alpha m_\alpha \tau_\alpha}(\mathbf{x}_1) \phi^*_{\beta m_\beta \tau_\beta}(\mathbf{x}_2) \phi^*_{\gamma m_\gamma \tau_\gamma}(\mathbf{x}_3) \, \hat{W} \\
&\times \; \big[\phi_{\mu m_\mu \tau_\mu}(\mathbf{x}_4) \phi_{\nu m_\nu \tau_\nu}(\mathbf{x}_5) \phi_{\lambda m_\lambda \tau_\lambda}(\mathbf{x}_6) \; - \; \phi_{\mu m_\mu \tau_\mu}(\mathbf{x}_4) \phi_{\nu m_\nu \tau_\nu}(\mathbf{x}_6) \phi_{\lambda m_\lambda \tau_\lambda}(\mathbf{x}_5) \\
&\;\; + \; \phi_{\mu m_\mu \tau_\mu}(\mathbf{x}_5) \phi_{\nu m_\nu \tau_\nu}(\mathbf{x}_6) \phi_{\lambda m_\lambda \tau_\lambda}(\mathbf{x}_4) \; - \; \phi_{\mu m_\mu \tau_\mu}(\mathbf{x}_5) \phi_{\nu m_\nu \tau_\nu}(\mathbf{x}_4) \phi_{\lambda m_\lambda \tau_\lambda}(\mathbf{x}_6) \\
&\;\; + \; \phi_{\mu m_\mu \tau_\mu}(\mathbf{x}_6) \phi_{\nu m_\nu \tau_\nu}(\mathbf{x}_4) \phi_{\lambda m_\lambda \tau_\lambda}(\mathbf{x}_5) \; - \; \phi_{\mu m_\mu \tau_\mu}(\mathbf{x}_6) \phi_{\nu m_\nu \tau_\nu}(\mathbf{x}_5) \phi_{\lambda m_\lambda \tau_\lambda}(\mathbf{x}_4) \big] \, .
\end{aligned}
\tag{3.2}
$$

The matrix elements calculated according to Eqs (3.1) are stored in an unformatted FORTRAN-type file in a precise order and truncated according to the maximum major-shell number for a single state (N_max), a pair (E^{12}_{max}) and a triplet (E^{123}_{max}) of spherical harmonic oscillator states. In principle, these orbits do not need to be harmonic oscillator states (for example, if a different basis set is used) but they have to be spherical single particle states and have analogous quantum numbers as the HO case.

One first order the single particle basis in increasing values of of the shell quandum number $N_a \equiv 2n_a + \ell_a$. For each value of N_a, orbits are ordered in increasing angular momentum ℓ_a and with total angular momentum $j^-_a = \ell_a - \frac{1}{2}$ coming before $j^+_a = \ell_a + \frac{1}{2}$.

i_α		N_α	ℓ_α	$2j_\alpha$	(n_α)	(π_α)
1	$\rightarrow$	0	0	1	0	0
2	$\rightarrow$	1	1	1	0	1
3	$\rightarrow$	1	1	3	0	1
4	$\rightarrow$	2	0	1	1	0
5	$\rightarrow$	2	2	3	0	0
6	$\rightarrow$	2	2	5	0	0
7	$\rightarrow$	3	1	1	1	1
8	$\rightarrow$	3	1	3	1	1
9	$\rightarrow$	3	3	5	0	1
10	$\rightarrow$	3	3	7	0	1
11	$\rightarrow$	4	0	1	2	0
12	$\rightarrow$	4	2	3	1	0
etc...						

(3.3)

Appendix A contains an example of FORTAN and C/C++ source code that can generate TBI files according to the conventions above.

3.1.2 One-Body Density Matrix

Appendix A

Code for generating TBI files

Software Licence