8. Program DBSR_HD

This stage of the DBSR complex deals with the final diagonalization of the Hamiltonian matrix. The eigenproblem (2.17) in the *B*-spline basis leads to a generalized eigenvalue problem (7.1). For the solution of this problem, we use the standard routines from the LAPACK library (http://www.netlib.org/lapack/index.html). If desired, the user can replace these routines with others.

8.1. Structure and data flow

The block diagram of the program DBSR_HD, along with the data flow, is given in Fig.8.1. First, the program reads the data common to all partial waves. This includes the description of the target states **(read_target_jj)**, the *B*-spline parameters (**read_knot_dat**), and the parameters of the calculations and the experimental thresholds if any (routine **read_arg**). Then the program executes fully independent calculations for each partial wave under consideration.

First, the program reads information from the **dbsr_mat.nnn** file about the basis used to represent the Hamiltonian and overlap matrixes. Recall that the Hamiltonian matrix is recorded in DBSR_MAT not in the pure *B*-spline basis but in the positive-energy basis of the channel eigenvalues. The corresponding transformation matrix is reading and checked in the subroutine **check_dbsr_mat**. The main calculations concerning the diagonalization of Hamiltonian matrix then proceeds in the subroutine **diag_mat** and consists the following steps. We fist reduce the generalized eigenvalue problem

$$\mathbf{HC} = \mathbf{SCE} \tag{8.1}$$

to the standard eigenvalue problem by a Cholesky factorization of the overlap matrix **S.** The factorization has the form

$$\mathbf{S} = \mathbf{L}\mathbf{L}^T \tag{8.2}$$

where **L** is the low-triangle matrix obtained with LAPACK program DPOTRF. The equation (8.1) then can be written in the form

$$\mathbf{L}^{-1}\mathbf{H}(\mathbf{L}^T)^{-1}(\mathbf{L}^T\mathbf{C}) = (\mathbf{L}^T\mathbf{C})\mathbf{E} \tag{8.3}$$

or

$$\tilde{\mathbf{H}}\tilde{\mathbf{C}} = \tilde{\mathbf{C}}\mathbf{E}, \tag{8.4}$$

where $\tilde{\mathbf{H}} = \mathbf{L}^{-1}\mathbf{H}(\mathbf{L}^T)^{-1}$ and $\tilde{\mathbf{C}} = (\mathbf{L}^T\mathbf{C})$. The matrix $\tilde{\mathbf{H}}$ is obtained with LAPACK program DSYGST. Now we are in the position to correct target energies if needed. Such corrections are frequently applied in the practical calculations to improve the accuracy of the final results. We just modify the Hamiltonian matrix as $\tilde{\mathbf{H}} \to \tilde{\mathbf{H}} + \mathbf{D}$, where $\mathbf{D}$ is the diagonal matrix with energy corrections. This

option is controlled by the input parameter **iexp**. If this parameter is greater 0, the program reads experimental threshold energies from the file **thresholds** and calls the routine **add_exp** for the corresponding modification of the interaction matrix (see Fig. 8.1). Note that this simple procedure cannot be directly applied to the original generalized eigenvalue problem (8.1).

Next step is diagonalization of the matrix $\tilde{\mathbf{H}}$ and determination of the matrix of solutions $\tilde{\mathbf{C}}$ and corresponding eigenvalues **E**. This is performed with LAPACK routine DSYEV. At this states we are also in the position to determine the weights of different channel functions and perturbers in the given solutions. They are defined as sum of square of the corresponding elements in the matrix $\tilde{\mathbf{C}}$. The weights can be used in the classifications of the solutions and in other applications. As indicated in scheme 8.1, The weights for different channels and (N+1)-electron configurations are determined in the subroutine **w_out** and saved in the file **w.nnn** for further possible usage. Finally, we perform back transformation and obtained the matrix of solutions **C**. This solutions are still in the basis of channel eigenvalues obtained in the DBSR_MAT stage.

After diagonalizing the Hamiltonian matrix, the results are treated in different ways, depending on the type of calculation specified by the input parameter **itype**, namely, scattering calculations, **itype**=0, photoionization calculations, **itype**=1, or bound-state calculations, **itype**= –1. For scattering calculations, including photoionization, the program first generates the **h.nnn** file in the subroutine **h_out**. This file has a standard structure, H.DAT, first adopted in the Belfast *R*-matrix codes. For a detailed description of this file, see section 8.4. Note that the surface amplitudes, which are needed to generate the final *R*-matrix, are directly defined in the present implementation by the expansion coefficients of the last *B*-spline, which is the only spline with nonzero value at the boundary. To find these coefficients, we additionally perform the transformation of solutions from the channel-eigenfunctions basis to the initial pure *B*-spline basis. It is done in the subroutine **rsol.out** (see scheme 8.1), which also records the full set of inner-region solutions in file **rsol.nnn** if **itype=**1. These data are then used by program DBSR_DMAT to generate the corresponding dipole matrix **d.nnn** for photoionization calculations by the DBST_PHOT program. An important difference from the earlier codes is that we do not provide any Buttle correction in the H.DAT file, considered the set of solutions to be effectively complete.

Another option in the BSR_HD program is a bound-state calculation. The resulting bound-state solutions are generated by the routine **b_out** and output in the **dbound.nnn** file (see scheme 8.1) in the original B-spline basis. In general, we may get a huge number of solutions, not all of which are physical or helpful in further calculations. A set of input parameters is therefore introduced in order to avoid the generation of large output with excess information. The parameter **msol** limits the total number of output solutions, whereas the parameters **Emin** and **Emax** restrict the solutions according to their energies. The

dbound.nnn file also contains the label for the given solution as the leading configuration in its expansion. The principal quantum numbers for bound-state solutions are defined according to the number of their nodes in the internal region. Information about the solutions stored in the unformatted **dbound.nnn** files user can gather in the separate file, using the utility **dbound_tab** (see section 12.x). Addiional information about the solutions (including the leading terms in the expansion) can be provided in the **dbsr_hd.nnn** files based on the parameter **cwt** (see subsection 8.4).

REMARKS:

1. Note that parameter **itype** should be consistent with the same parameter used in the DBSR_MAT for calculation of channel-orbital basis because it also defines the applied boundary conditions. It means that if **itype** $\geq$ 0 in DBSR_MAT, it is also should be 0 or 1 in DBSR_HD, and if **itype**=-1 in DBSR_MAT, the same values should be used in DBSR_HD. It differs the DBSR procedure from the BSR algorithm and is related to the fact that we use the channel-orbital basis (obtained from the diagonalization of channels blocks) to eliminate the negative part of spectrum (none-pare approximation). It also considerably reduces the memory required to record the total Hamiltonian matrix
2. Another specific feature of the DBSR_HD (or BSR_HD) algorithm is that corrections to the target energies are allowed to change the order of target states. In this case, the **h_out** subroutine is replaced by the **h_out_exp** which records new target and channels order in the output H.DAT file. The further using the H.DAT file in different applications, the user may needs new **target** file with new order of target states and related scattering channels. This corrected **target** file can be obtained with utility-programs **h_targb** or **convert_target_jj**.
3. Fig. 8.2 provide typical example of the **dbsr_hd.001** file. The output contains main parameters of the case, in particular, the output contains the first five eigen-energies, Eval. If they are unphysically too small, it indicate the problems, connected first of all with overlap matrix. In this case, the user should check the warnings, if any, about the big overlap matrix elements. In the given example, there is no problems with energies, however, it is worth to eliminate this big overlaps by imposing additional orthogonal conditions. In the example, we need to impose additional orthogonal condition for the ksV orbital to the one from the substitution list. To find this orbital, the user should check the ns substitution orbitals for the target 11 or 18, and then add the corresponding orthogonal condition to the end of the **cfg.001** and re-run DBSR_MAT and DBSR_HD for the given partial wave.

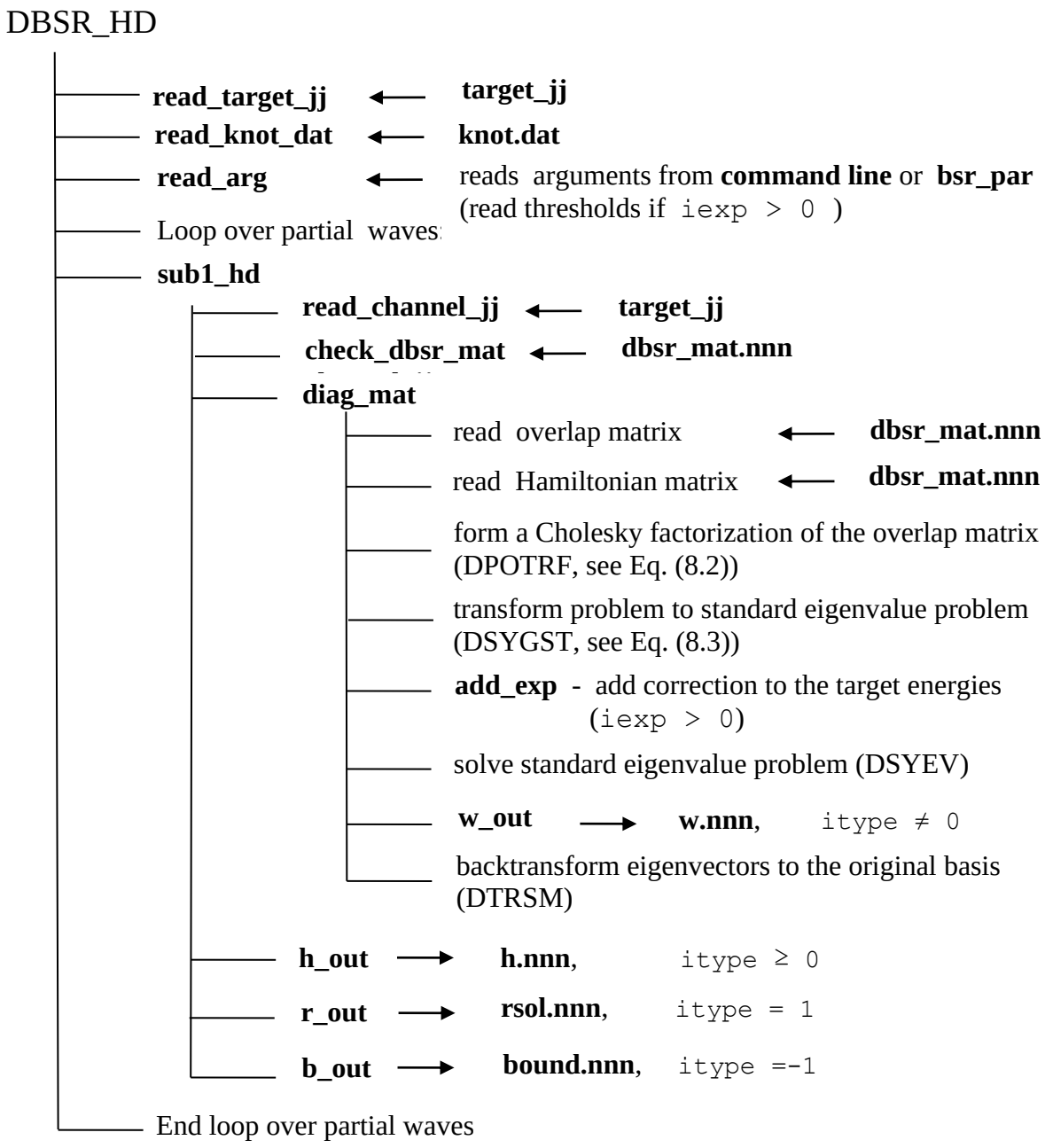

Fig. 8.1. Block diagram for the program DBSR_HD and data flow (see text).

8.3. Data files

The following is a summary of input and output data files used by DBSR_HD. The unit numbers and file names are defined within the program. Some files have an extension **nnn** that depends upon the partial wave under consideration.

dbsr_par File type: formatted sequential input.
Written by user.
Read by routine **read_arg.**
Description: input parameters for the given run.

target_jj File type: formatted sequential input.
Prepared by user and modified by programs DBSR_PREP and DBSR_CONF.
Read by routine **read_target_jj.**
Description: contains the description of the target states and scattering channels.

knot.dat File type: formatted sequential input.
Written by user.
Read by routine **read_knot_dat** from the DBS library.
Description: input parameters that define the *B*-spline grid.

cfg.nnn File type: formatted sequential input
Created by program DBSR_CONF.
Read by routine **read_conf_jj.**
Description: list of configurations in spectroscopic notation for the partial wave **nnn**.

target.bsw File type: unformatted sequential input.
Created by program DBSR_PREP.
Read by routine **read_dbsw.**
Description: target one-electron orbitals in the *B*-spline basis.

dbsr_mat.nnn File type: unformatted sequential input.
Created by program DBSR_MAT.
Read by routine **read_bsrmat**.
Description: Hamiltonian and overlap matrixes, and the asymptotic coefficients for the given partial wave.

thresholds File type: formatted sequential input.
Prepared by user.
Read by routine **read_arg.**
Description: experiment target energies, optional.

dbsr_hd.nnn File type: formatted sequential output.
Written by program BSR_HD.
Read by user.
Description: running information.

h.nnn File type: unformatted sequential output.
Created by routine **h_out.**
Read by program FARM, STGF, BSR_PHOT or other programs that deal with the outer region**.**
Description: channel information and the surface amplitudes for solutions in the

inner region.

dbound.nnn File type: unformatted sequential output.
Created by routine **b_out.**
Read by programs BSR_DMAT, DBSR_POL, DBSR_ZF, dbound_bsw.
Description: bound-state solutions for given partial wave in the *B*-spline representation (8.2).

rsol.nnn File type: unformatted sequential output.
Created by routine **r_out.**
Read by program BSR_DMAT.
Description: full set of *R*-matrix solutions in the inner region.

w.nnn File type: unformatted sequential output.
Created by routine **w_out.**
Read by routine **b_out** and program BSR_PHOT.
Description: weights of different channels in the inner-region solutions. Used for classification and sorting of bound-state solutions.

8.4. Input data

Input data can be provided in the command line or in the input file **dbsr_par** (the data from the command line overwrite the data from the input file). All data have the default values indicated in the brackets. All default values, along with unit numbers and default file names, are placed in the module **dbsr_hd**.

klsp1[1] first partial wave under consideration.

klsp2[klsp1] last partial wave under consideration.

klsp[1] the specific partial wave to be considered.

itype[0] mode of calculations:
itype= 0 - scattering calculations; the **h.nnn** file is generated.
itype= 1 - photoionization calculations; **rsol.nnn** file is generated additionally.
itype=-1 – bound-state calculations; **dbound.nnn** file is generated.

iexp[0] controls the treatment of experimental thresholds (see text).

msol[0] restricts the number of bound-state solutions for output in **dbound.nnn** file. msol=0 means output of all solutions.

Emin[0] only solutions with $E >$ Emin are considered (Emin=0 means no restrictions)

Emax[0] only solutions with $E <$ Emax are considered (Emax=0 means no restrictions).

cwt[0] if > 0, the expansions for the first msol solutions will be output in the **dbsr_hf.nnn** file for the components with coefficients > cwt.

iwt[0] if > 0, the file **w.nnn** with weights of solution is created.

8.5. Structure of the H.DAT file

We keep the same structure for the output H.DAT files as in the Belfast *R*-matrix codes [4]. The **h.nnn** files from the DBSR_HD program contain information only for one partial wave. In order to generate the full H.DAT file, the user should run the utility program **sum_hh** (see section 13). Below is a summary of the output records in the **h.nnn** files.

1. nelc,nz,lrange,km,ntarg,RA,BSTO

nelc – number of electrons in the target
nz – atomic number
lrange – maximum value of (l+1)-value for the continuum orbitals
km – maximum multipole index for the asymptotic coefficients
RA – *R*-matrix radius
RB – the value of the logarithmic derivative to be imposed on the continuum orbitals
The variables **lrange** and **RB** were included only for consistency with earlier versions

2. etarg (1:ntarg) – target energies, in a.u.
3. jtarg (1:ntarg) – 2J –values for the target states
4. (0,i=1,ntarg)) – no spin information for *jj*-coupling

5. (0.d0,0.d0,0.d0,i=1,lrange)

This record in the standard H.DAT file should contain the Buttle corrections, which are not used in the present implementation.

6. jpar,0,ipar,nch,nhm,more

jpar – 2J –value for the given partial wave
par – total parity for the given partial wave (0|1)
nch – number of scattering channels
khm – number of inner-region solutions
more – if = 0, the current partial wave id final

7. NCONAT(1:ntarg) – the number of channels associated with each target state
8. lch(1:nch), kch(1:nch) – channel orbitals angular momenta and κ-values
9. CF(1:nch,1:nch,1:mk) – matrix of asymptotic coefficients (2.31)
10. eval(1:nhm) – *R*-matrix poles in descending order (values E_k in eq. (2.17))
11. wmat(1:nch,1:nhm) – corresponding surface amplitudes (values P_{ik} in eq. (2.17))

8.6. Output of the R-matrix solutions

Summary of the output records in the **rsol**.**nnn** files.

1. mhm,khm,nch,npert,ms,nsp,nsq
2. eval(1:khm)
3. a(1:nhm,.) → repeat **khm** times

mhm – dimension of the solution vector in the *B*-spline basis (ms * nch + npert).
khm – number of solutions.
nch – number of channels.
npert – number of bound (*N*+1)-electron states.
ms – number of *B*-splines for each channel.
nsp – number of *B*-splines for the large component.
nsq – number of *B*-splines for the small component.
eval(.) – *R*-matrix eigenvalues.
a(1:mhm,.) – corresponding eigenvectors.

8.7. Output of bound solutions.

Summary of the output records in the **dbound.nnn** files.

1. mhm,nch,npert,ns,jpar,ipar,nbound
2. i,LABEL
3. eval(i),Ebind(i)
4. a(1:mhm,i)
repeat records 2-4 `nbound` times for each bound state *i* .

mhm – dimension of the solution vector in the *B*-spline basis.
nch – number of channels.
npert – number of (*N*+1)-electron bound channels.
ns – number of *B*-splines.
jpar – 2*J* –value for the given partial wave.
ipar – total parity for the given partial wave, (-1|1).
nbound - number of solutions.
LABEL – spectroscopic notation for solution *i*.
eval – energy of the solution in a.u.
Ebind – binding energy.
a – solution vectors in the *B*-spline basis, see Eq. (7.4).

8.8. Output of weights.

Summary of the output records in the **w.nnn** files.

1. nch, npert, nsol
2. W(1:nch+npert)

repeat record 2 for each solution **nsol** times.

nch – number of channels.
npert – number of (N+1)-electron configurations.
nsol – number of solutions.
W – array of weights for each channel and (N+1)-electron perturbers.

8.9.*MPI version – DBSR_HD_MPI*

The MPI version DBSR_HD_MPI is based on the BLACS and SCALAPACK libraries. The BLACS (Basic Linear Algebra Communication Subprograms, www.netlib.org/blacs) is a linear algebra oriented message passing interface that can be implemented across a large range of distributed memory platforms. In particular, BLACS are used as the communication layer for the SCALAPACK project (www.netlib.org/scalapack), which involves implementing the LAPACK library on distribute memory machines. The BACS and SCALAPACK projects are used two-dimensional block-cyclic distribution, Fig 8.3. The main parameters which we will use to describe the particular distribution are the size of block, **nblock**, number of rows, **p**, and number of columns, **q**, in the distributed matrices, presented in the right part of figure. The colored block represented part of distributed matrix connected with one processor. Overall, number of processors involved is **p×q**.

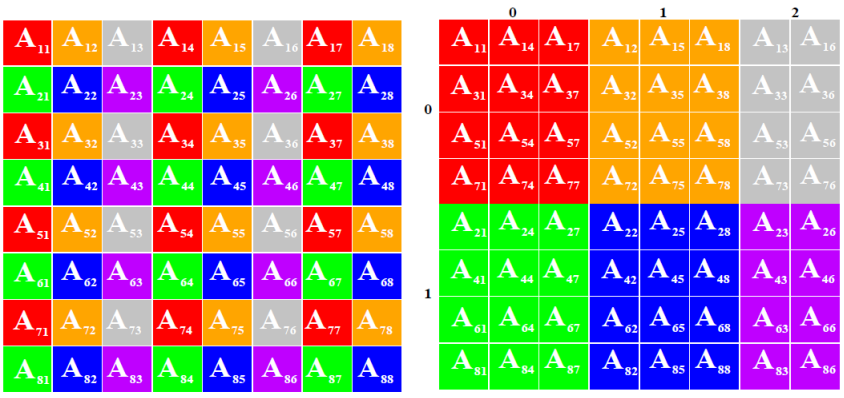

Fig.8.3. Two-dimensional block-cyclic distribution.

Overall, the DBSR_HD_MPI program follows the same logic as serial version DBSR_HD, however, all main matrixes are distributed over p*q processor. The block scheme of the DBSR_HD program is given in Fig. 8.4. All MPI operation are performed through the BLACS routines, and all read/write operations are governed by the master processor only. In Fif.8.4 we use blue color for subroutines which are run only on the master processor. The SCALAPACK routines have the same names but with first letter P. The new SCALAPACK routine PDGEADD turned out to be very useful to fill in and transform of the distributed arrays.