3. General features of the DBSR complex

3.1. Preparation of target states.

In the present implementation, the preparation of the target states is the responsibility of the user. The input target description includes both the target orbitals and configuration expansions. The present complex was designed, first of all, for a detailed and accurate study of the low-energy electron/photon scattering from neutral atoms, where the accuracy of target states is equally important as the scattering model. Especially it concerns the study of low-energy resonance structures. There is a set of different atomic-structure programs which can be used for generating the target states in the fully-relativistic approximation. The most popular is the GRASP complex (Jönsson et al 2007, 2013) which was developed for many years by different authors. Originally, the DBSR complex was oriented to obtain the target description from the GRASP calculations and as a basic representation for the different atomic states we choose the format of the GRASP package. In particular, the configuration expansion for specific atomic state is given in a so-called c-file, which is a list of configurations in the spectroscopic notation along with the relevant expansion coefficients. The corresponding one-electron radial functions should be provided in the *B*-spline representation (we will call such files as bsw-files). These files can be generated from the corresponding files in the GRASP package as a preliminary step of calculations. Hence, each input target state in the present package should be represented by a pair: **name.c** for the configuration expansion and **name.bsw** for the relevant one-electron radial functions.

This preparation of the target states looks like a serious complication in comparison with the DARC code, where one can simply provide a single set of proper one-electron orbitals. On the other hand, the present method allows for a separate optimization of different target states, or even to use non-orthogonal orbitals for one target state. It thereby enables the user to generate accurate target states with a relatively small number of configurations and directly include important effects such as term-dependence and relaxation. The generation of the bsw-files for the one-electron radial functions frees us from the use of a specific atomic-structure package. Furthermore, the user chooses the appropriate *R*-matrix radius at this stage and can perform additional checks of the target states.

Last years the DBSR complex was extended by including such structure programs as DBSR_CI, DBSR_HF and DBSR_MCHF, which are directly working with B-spline representation for radial functions. In the most cases, these programs are enough to generate the all needed atomic states with desirable accuracy. It makes the DBSR complex to be self-sufficient and it also simplifies for user to carried out the calculations.

3.2. Main programs and data files

The program modules are run sequentially as outlined in Fig. 3.1. Besides a set of c- and bsw-files for the target description, the user should prepare three input files (**knot.dat, target, dbsr_par**), which contain the basic

information about the given run and are read by all programs. **knot.dat** contains the *B*-spline grid and should be created at the time of the target preparation (a detailed description of **knot.dat** is given in section 12.1). **target_jj** contains the description of all target states and partial waves for the given run (see program DBSR_PREP and DBSR_CONF for details). **dbsr_par** defines the input parameters which control the execution of the programs. All input parameters in **dbsr_par** have the format **name=value**, and **name** should be placed at the beginning of the line; otherwise the corresponding assignment will be ignored. This format is similar to the FORTRAN NAMELIST, but it does not require specific NAMELISTs and is rather convenient in practical calculations. Furthermore, all input data form **dbsr_par** can be alternatively provided in the command line, in the same format **name=value**. In this case the data from the command line overwrite the data from the input file. Data specific for a given partial wave are more convenient to be provided in the command line, while data common for all partial waves should be provided in the input file.

Other disk files, outlined in Fig.3.1. in bold, are produced by programs at each stage and are normally deleted after use. The final output files are the H and D files produced at the end of DBSR_HD and DBSR_DMAT programs. They contain all the information from the internal region that is required for the external region codes to run scattering or photoionization calculations. The present package only deals with the internal region, and the user then can then choose the program for the treatment of the external region. In the case of photoionization calculations, two additional programs, DBSR_MULT and BSR_DMAT, should be run, in order to treat the dipole operator. The photoionization cross sections can then be produced by the program BSR_PHOT. This program, however, is based on the differential solver of Crees (1981). While very general, it is not the fastest program available.

Another specific feature of the present file system is that the intermediate files are generated for each partial wave individually. Such files have an extension which depends on the partial wave under consideration. For example, **cfg.001**, **cfg.002,** etc., are the configuration files for the first, second and further partial waves. Thereafter, the files which are generated for the specific partial wave will be denoted by **nnn** in the extension. Such a file system sometimes generates a large number of intermediate files, but allows us to run the calculations for different partial waves in parallel. It is especially convenient when the computer has several processors.

3.3. Others features

The BSR package is written in FORTRAN 95 and uses double precision. It is designed to be transportable and has been implemented on different computers and operating systems. All unit numbers and file names are defined within the program. They are collected in the corresponding modules, along with all other parameters using in the specific program. Hence, if the user wants to change the default values for some parameter, it only needs to be done in one place.

All principal arrays in the DBSR package are allocatable, and they are initialized dynamically according to the input parameters. Consequently, the user should not worry about the dimensions and their correspondence to the input parameters. On the other hand, it limits the complexity of the physical model (e.g. number of scattering channels, number of orbitals, and so on) which we can treat at the given computer by the size of the RAM memory. (Virtual memory will not help much here.)

The DBSR package is organized as a set of main programs, shown in Fig.3.1, utility programs, which help the user to prepare and process the input/output data, and four libraries DBS, ZCONFJJ, ZCOM and LAPACK, which contain routines common to all programs. The DBS library, one of the most important parts of the package, contains routines for using B-splines in the atomic structure calculations. The ZCONFJJ library contains routines that deal with the configuration and close-coupling expansions, and ZCOM contains routines for a variety of auxiliary computations that are common for different atomic-structure calculations. The LAPACK library (http://www.netlib.org/lapack/index.html) was chosen to provide the linear-algebra calculations, such as matrix diagonalization or solving a system of linear equations. For the compilation of the DBSR program, a UNIX/LINUX user can use the makefiles provided along with the source files. Other users can use these files as a guide for compilation in a specific environment.

The main most time-consuming programs have the MPI versions. It allows users run the large-scaled calculations, especially in the RMPS approach.

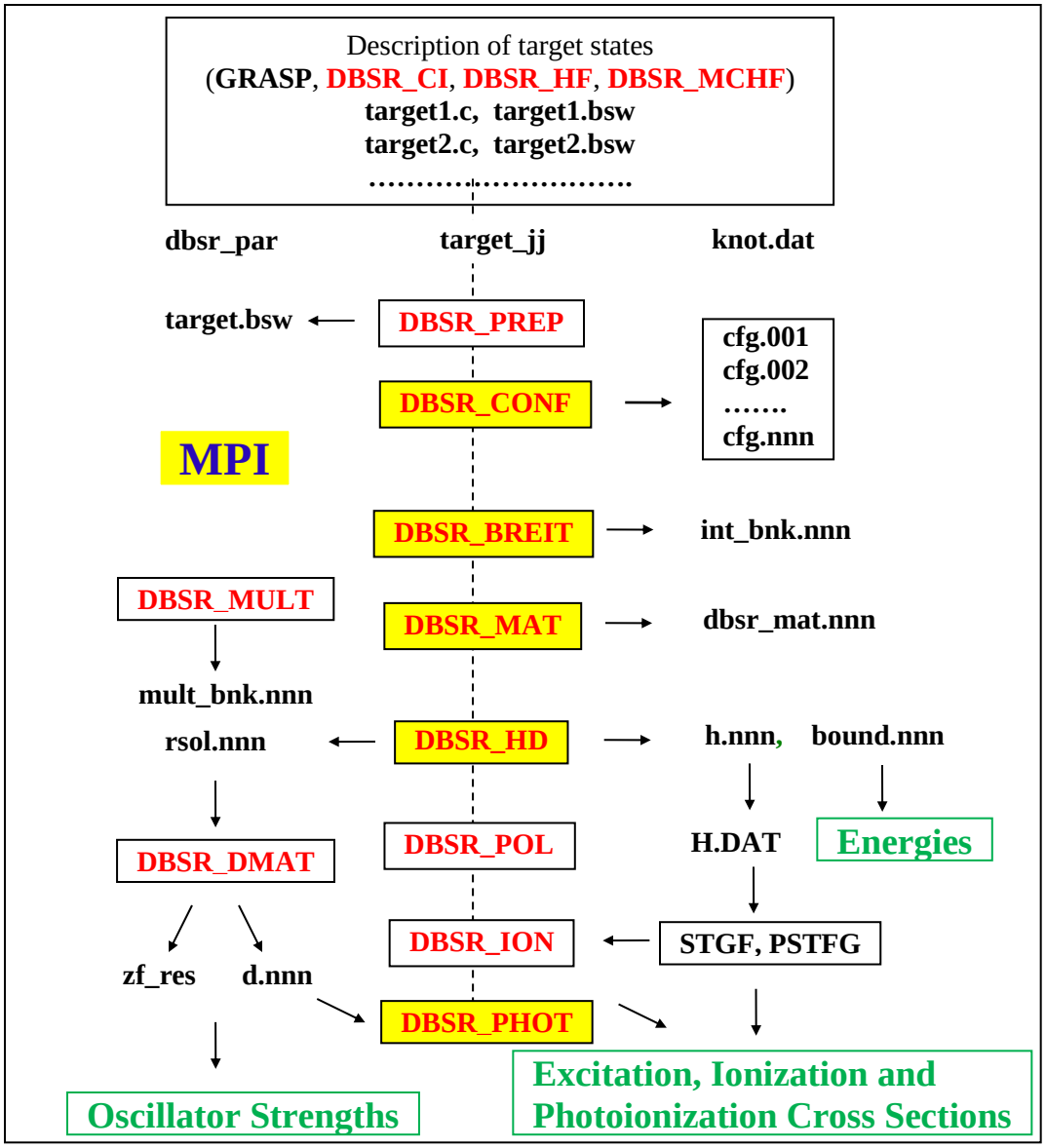

Fig. 3.1. Structure of the DBSR package. Notation: capital names stand for the programs; the vertical dashed line indicates the sequential execution of the programs, from top to bottom; bolded names stand for the most important intermediate data files.