Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Physics Communications 174 (2006) 273–356

Computer Physics Communications

www.elsevier.com/locate/cpc

BSR: B-spline atomic R-matrix codes ☆

Oleg Zatsarinny

Department of Physics and Astronomy, Drake University, Des Moines, IA 50311, USA

Received 19 April 2005; received in revised form 26 October 2005; accepted 30 October 2005

Available online 20 December 2005

Abstract

BSR is a general program to calculate atomic continuum processes using the B-spline R-matrix method, including electron–atom and electron–ion scattering, and radiative processes such as bound–bound transitions, photoionization and polarizabilities. The calculations can be performed in LS-coupling or in an intermediate-coupling scheme by including terms of the Breit–Pauli Hamiltonian.

New version program summary

Title of program: BSR
Catalogue identifier: ADWY
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/ADWY
Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland
Computers on which the program has been tested: Microway Beowulf cluster; Compaq Beowulf cluster; DEC Alpha workstation; DELL PC
Operating systems under which the new version has been tested: UNIX, Windows XP
Programming language used: FORTRAN 95
Memory required to execute with typical data: Typically 256–512 Mwords. Since all the principal dimensions are allocatable, the available memory defines the maximum complexity of the problem
No. of bits in a word: 8
No. of processors used: 1
Has the code been vectorized or parallelized?: no
No. of lines in distributed program, including test data, etc.: 69 943
No. of bytes in distributed program, including test data, etc.: 746 450
Peripherals used: scratch disk store; permanent disk store
Distribution format: tar.gz
Nature of physical problem: This program uses the R-matrix method to calculate electron–atom and electron–ion collision processes, with options to calculate radiative data, photoionization, etc. The calculations can be performed in LS-coupling or in an intermediate-coupling scheme, with options to include Breit–Pauli terms in the Hamiltonian.
Method of solution: The R-matrix method is used [P.G. Burke, K.A. Berrington, Atomic and Molecular Processes: An R-Matrix Approach, IOP Publishing, Bristol, 1993; P.G. Burke, W.D. Robb, Adv. At. Mol. Phys. 11 (1975) 143; K.A. Berrington, W.B. Eissner, P.H. Norrington, Comput. Phys. Comm. 92 (1995) 290].

PACS: 34.80.Dp; 34.80.Kw; 32.80.Fb; 31.15.Ar

Keywords: Electron–atom scattering; Electron–ion scattering; Photoionization; Oscillator strengths; Polarizability; R-matrix; B-splines

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

E-mail address: oleg_zoi@yahoo.com (O. Zatsarinny).

0010-4655/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2005.10.006

Contents

1. Introduction . . . 275
2. General theory . . . 277
   2.1. The close-coupling expansion . . . 277
   2.2. The R-matrix method . . . 279
   2.3. The external region . . . 281
   2.4. Radiative processes . . . 282
   2.5. Bound-state calculations . . . 283
   2.6. Relativistic corrections . . . 284
3. General features of the BSR complex . . . 286
   3.1. Preparation of target states . . . 286
   3.2. Main programs and data files . . . 286
   3.3. Others features . . . 286
4. Program BSR_PREP . . . 287
   4.1. Outline of the BSR_PREP calculations . . . 287
   4.2. Data files . . . 288
   4.3. Input data in file **target** . . . 289
5. Program BSR_CONF . . . 289
   5.1. Outline of the BSR_CONF calculations . . . 289
   5.2. Data files . . . 292
   5.3. Input parameters . . . 293
   5.4. Modification of file **target** . . . 293
6. Program BSR_BREIT . . . 294
   6.1. Outline of the BSR_BREIT calculations . . . 294
   6.2. Structure and data flow . . . 295
   6.3. Data files . . . 297
   6.4. Input parameters . . . 297
   6.5. Structure of the angular-coefficients databank . . . 297
7. Program BSR_MAT . . . 298
   7.1. Outline of the BSR_MAT calculation . . . 298
   7.2. Structure and data flow . . . 301
   7.3. Data files . . . 302
   7.4. Input data . . . 303
   7.5. Output of Hamiltonian matrices and asymptotic coefficients . . . 304
8. Program BSR_HD . . . 305
   8.1. Structure and data flow . . . 305
   8.2. Data files . . . 307
   8.3. Input data . . . 308
   8.4. Structure of the H file . . . 309
   8.5. Output of the R-matrix solutions . . . 309
   8.6. Output of bound solutions . . . 310
   8.7. Output of weights . . . 310
9. Program MULT . . . 310
   9.1. Outline of the MULT calculations . . . 310
   9.2. Structure and data flow . . . 313
   9.3. Data files . . . 315
   9.4. Input parameters . . . 315
   9.5. Structure of the dipole databank . . . 315
10. Program BSR_DMAT . . . 316
   10.1. Dipole transition matrix . . . 316
   10.2. Oscillator strengths and decay probabilities . . . 318
   10.3. Structure and data flow . . . 318
   10.4. Data files . . . 320
   10.5. Input parameters . . . 321
   10.6. Output of dipole matrix . . . 321
   10.7. Output of radiative data in **zf_res** . . . 321
11. The program BSR_PHOT . . . 322
   11.1. Structure and data flow . . . 322
   11.2. Data files . . . 323
   11.3. Input parameters . . . 324
12. Libraries . . . 324

12.1. Description of B-splines . . . 325
12.2. General routines . . . 327
12.3. Atomic orbital routines . . . 331
12.4. Evaluation of two-electron integrals . . . 332
12.5. The library ZCONF . . . 343
12.6. The library ZCOM . . . 346
13. Additional programs-utilities . . . 347
14. Compilation and test runs . . . 349
14.1. Compilation . . . 349
14.2. Test runs . . . 349
14.3. LS-coupling test run . . . 350
14.4. jK-coupling test run . . . 351
14.5. Photoionization calculations . . . 352
14.6. Bound-state calculations . . . 353
Acknowledgements . . . 355
References . . . 355

1. Introduction

Collisions of electrons and photons with atoms and ions are among the most elementary processes that can be studied within the framework of quantum mechanics. The latter processes include elastic scattering, inelastic excitation, and ultimately collisional break-up in ionization. These collisions determine to a considerable extent the energy balance for different plasma and laser devices, while the understanding of the underlying mechanisms allows one to study comprehensively the properties of atomic systems and their structure. Benchmark experiments and calculations have further advanced our basic knowledge of this field, while at the same time providing urgently needed input data for modeling programs describing astrophysical and laboratory plasmas, planetary atmospheres, and lasers. Due to the many difficulties associated with experimental investigations of these collisions, such as the normalization of absolute cross sections and/or low count rates, and also due to the enormous amount of data needed for the modeling programs, the vast majority of currently available input data for these programs consists of theoretical predictions.

During the past decade, a number of general computer codes have been developed to generate accurate data for the various scattering parameters. The R-matrix technique describing electron and photon scattering in complex atoms and ions was introduced to atomic collision physics by Burke and collaborators [1,2] and now is the method most frequently applied for the determination of various atomic properties. The compilation volume by Burke and Berrington [3] includes the major papers in R-matrix theory until a decade ago, along with a comprehensive list of calculations using the method. Two of the major world-wide projects successfully undertaken with extensive use of the general R-matrix package published by Berrington et al. [4] are the Opacity Project (http://heasarc.gsfc.nasa.gov/topbase/op.html) and the Iron Project (http://www.usm.uni-muenchen.de/people/ip/iron-project.html). The above suite of codes, RMATRX-I, still uses the basic ansatz of the version first published in 1974 [5]. While many subroutines were rewritten to increase efficiency and user-friendliness [6] and to include relativistic effects [7], these were essentially updates to the original version. More radical changes were made in RMATRX-II, with particular emphasis placed on improving the efficiency of the angular-momentum package. The latter code is now parallelized and resulted in the recent publication of PRMAT [8]. Straightforward applications of this program, however, are more limited than those of RMATRX-I; currently only electron collisions (no photon processes) can be handled within a non-relativistic framework. Consequently, work on producing a robust parallel version of RMATRX-I is still in progress (http://www.osti.gov/scidac/fes/projects/pindzola.html). Relativistic effects can be included in RMATRX-I at the level of first-order perturbation theory by adding the one-body relativistic terms of the Breit–Pauli Hamiltonian (essentially the Pauli approximation) to the non-relativistic Hamiltonian that needs to be diagonalized. Although fully relativistic versions of both atomic structure and collision packages exist (GRASP [9] and DARC [10], respectively), applications of DARC have been comparatively rare and mostly restricted to highly ionized systems.

There are also other packages around, such as the published code RMATRX-ION [11] to handle general ionization as well as ionization–excitation processes, another modified version of RMATRX-I (http://amdpp.phys.strath.ac.uk/rmatrix/), and the intermediate-energy R-matrix code IERM [12] based on the theory outlined by Burke et al. [13]. In addition, many other researchers developed their own codes, but these are generally unpublished and the developers were mostly interested in special applications, particularly (quasi)-two electron problems such as electron scattering from alkali-like targets.

One of the principal ingredients of the above-mentioned programs is the usage of one set of orthogonal one-electron orbitals: one subset is used to construct the target states in multi-configuration expansions and the other one is employed to represent the scattered projectile. It simplifies the calculations and allows one to develop fast effective programs. On the other hand, it leads to three major problems that often arise when the current suite of R-matrix codes is applied to truly complex targets. These problems include (i) the difficulties in describing all target states of interest for a given calculation to sufficient accuracy, (ii) the likely occurrence of unphysical structures, so-called "pseudo-resonances", when an attempt is made to address the former problem, and (iii) numerical difficulties due to an ill-conditioned orthogonalization procedure and the need to modify the so-called "Buttle correction".

Given the success of the R-matrix method for atomic collision processes, it is not surprising that many modifications of the R-matrix codes and more detailed formulation of the algorithms have been suggested since the programs were first published. They include but are not limited to the eigenchannel formulation [14], different choices of basis functions [15], particularly targeted to eliminate or to reduce the sometimes problematic Buttle correction [16–18], or recipes to avoid the pseudoresonance problem that occurs in many sophisticated applications [19,20].

The present program is another implementation of the R-matrix method with two significant innovations compared to the existing codes:

- non-orthogonal orbitals can be used to represent both the bound and continuum one-electron orbitals;
- a set of B-splines defines the R-matrix basis functions.

The use of non-orthogonal bound orbitals generally allows us to achieve a much higher accuracy in the description of the target states. Now they can be optimized in separate calculations, and high level of accuracy can be achieved with compact configuration expansions. The use of non-orthogonal continuum orbitals can drastically reduce the pseudo-resonance problem. Indeed, in standard applications of the R-matrix method, certain $(N+1)$-electron bound configurations must be included in the close-coupling expansion in order to compensate for orthogonality constraints imposed on the continuum orbitals. However, it can be difficult to keep the expansion fully consistent, and any inconsistency may lead to an undesirable pseudo-resonance structure. The use of a non-orthogonal orbital set allows us to avoid the introduction of these $(N+1)$-electron terms completely.

It follows from the above discussion that the non-orthogonal orbital technique may indeed overcome some of the major problems associated with the standard application of the R-matrix method. The use of non-orthogonal orbitals was so far avoided because of the fact that the most time-consuming part of atomic structure calculations is connected with the angular integration to construct the Hamiltonian matrix elements. When a set of efficient general codes had been developed for this task for fully orthogonal orbitals [21,22] and with some restricted non-orthogonality [23], it became possible to automate to a large extent this part of the inner-region problem in conventional R-matrix calculations. This resulted in a wide application of this method. A key step for the present program is the recent appearance of efficient codes for dealing with non-orthogonal orbitals in a general way [24,25]. Hence, it is now possible to use any non-orthogonal basis in the atomic structure calculations.

The choice of B-splines as basis functions has some advantages. They were introduced to atomic structure calculations in the 1980s and since became widely used due to their excellent numerical approximation properties (see, for example, the review by Bachau et al. [26]). The B-splines are bell-shaped piecewise polynomial functions of order k_s (degree $k_s - 1$), defined by a given set of points in some finite radial interval. Since the characteristic feature of the B-spline method is the solution of the Schrödinger equation within a box, which is very similar to the inner region in the R-matrix method, it can be expected that B-splines will be very effective in forming the R-matrix basis. Another important property of B-splines is that they form an effectively complete basis on the interval spanned by the knot sequence. This completeness of the B-spline basis ensures that no Buttle correction to the R-matrix elements is required. The use of B-splines as the R-matrix basis set was first outlined by van der Hart [27], who applied the method to e–H scattering and obtained excellent agreement with existing benchmark results.

As before, the final step in the calculation requires the solution of the scattering problem in the external region and to match the solutions on the boundary $r = a$. The present program deals only with the inner-region problem. Note, however, that once the R-matrix has been constructed, no further changes are required. Any improvements achieved for solving the outer-region problem, for example, through packages such as STGF (Seaton, unpublished) or FARM [28], can therefore be taken advantage of immediately in the present approach as well. Also, reconstructing the wavefunction, as needed for calculations of photoionization and photodetachment processes, proceeds along the same lines as in the standard implementation. The present programs can also be used for bound-states calculations in the B-spline basis.

The atomic R-matrix programs described here have been successfully applied to several problems. First, Zatsarinny and Froese Fischer [29] outlined a general B-spline R-matrix approach and demonstrated a successful implementation by treating photoionization of lithium. Zatsarinny and Tayal [30,31] then applied the method to electron scattering from sulfur and oxygen, and Zatsarinny et al. [32] achieved excellent agreement with experimental benchmark data for photodetachment of the $\mathrm{He}^-(1s2s2p)\,^4P^o$ metastable state and the 1s-photodetachment of C^- and B^- [33]. These calculations were performed as a proof-of-principle to show a working prototype version. While the results are most impressive, they could be (and in some cases were) also generated with other formulations of the R-matrix method, including the current Belfast implementation. During the past two years, the present code was extensively tested for truly complex systems, including electron scattering from the noble gases [34,35], where considerable improvement in comparison to results of standard R-matrix calculations was obtained.

This write-up is organized as follows. In the next section we will briefly summarize the most important aspects of the R-matrix theory. The general features of the present BSR complex are given in Section 3, while each program in BSR is then described in detail in Sections 4–11. Section 12 presents the libraries, common to all programs, including the B-spline library BSPLINE, which is the key part of the present implementation. Additional program utilities, which can be used for preparing and processing input/output data, are given in Section 13, and finally Section 14 describes the test runs.

Unless otherwise specified, all quantities are expressed in atomic units.

2. General theory

2.1. The close-coupling expansion

The problem of low-energy electron scattering from N-electronic atomic target is reduced to solving the time-independent Schrödinger equation

$$(H-E)\Psi_\alpha(\Gamma X, x_{N+1}) = 0 \tag{2.1}$$

with appropriate boundary conditions. The collision wave function $\Psi_\alpha(\Gamma X, x_{N+1})$ represents a fully antisymmetrized wave function of the system "target atom + projectile electron", where $X \equiv (x_1, x_2, \ldots, x_N)$, $x_i \equiv (\boldsymbol{r}_i, \sigma_i)$ with spatial $(\boldsymbol{r}_i)$ and spin (σ_i) coordinates of the ith electron, Γ is a complete set of quantum numbers of the $(N+1)$-electron system, and E is the total energy. The subscript α characterizes the initial conditions and usually denotes the incoming scattering channel.

The Hamiltonian H_{N+1} which describes the scattering of an electron on an N-electron atomic target with the nuclear charge Z has the form

$$H_{N+1} = \sum_{i=1}^{N+1}\left(-\frac{1}{2}\nabla_i^2 - \frac{Z}{r_i}\right) + \sum_{i>j=1}^{N+1}\frac{1}{r_{ij}}, \tag{2.2}$$

where $r_{i,j} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$, $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ being the vector coordinates of electrons i and j. The origin of the coordinate frame is set at the target nucleus, which is assumed to have an infinite mass. For the time being we have neglected all relativistic effects. We introduce a set of target eigenstates, and possibly pseudostates Φ_i, and their corresponding eigenenergies E_i by the equation

$$\langle\Phi_i|H_N|\Phi_j\rangle = E_i(Z,N)\delta_{ij}, \tag{2.3}$$

where the integration is carried out over all the space and spin coordinates of the target electrons. Then the total energy is $E = E_i + k_i^2/2$, with E_i being the energy of the target in the state i while $k_i^2/2$, represents the kinetic energy of the projectile electron. The target states Φ_i are expanded in terms of single-configuration basis states φ_j by

$$\Phi_i(x_1,\ldots,x_N) = \sum_j \varphi_j(x_1,\ldots,x_N)c_{ij}, \tag{2.4}$$

where the coefficients c_{ij} are determined by diagonalizing the target Hamiltonian. The basis configurations φ_j are constructed from a one-electron bound orbital basis, usually consisting of physical self-consistent field orbitals plus possibly additional pseudo-orbitals. The latter are included to represent correlation effects. Note that we do not assume the one-electron basis to be orthogonal, as usually imposed in scattering calculations. This allows us to optimize the bound orbitals in independent calculations for each target state and to use term-dependent one-electron radial functions.

The solution of (2.1) has to satisfy the boundary conditions of an incoming wave in some scattering channel α and outgoing waves in this and all other channels. In the close-coupling approximation, this solution is expanded in terms of a complete set of N-electron target wave functions Φ_i. The corresponding expansion coefficients play effectively the role of wave functions for the incident electron. In practice, one uses

$$\Psi_\alpha^\Gamma(x_1,\ldots,x_{N+1}) = \hat{A}\sum_{i=1}^{n}\bar{\Phi}_i^\Gamma\big(x_1,\ldots,x_N;\hat{r}_{N+1}\sigma_{N+1}\big)\frac{1}{r_{N+1}}F_{i\alpha}(r_{N+1}) + \sum_{j=1}^{m}c_j\chi_j^\Gamma(x_1,\ldots,x_{N+1}), \tag{2.5}$$

where $\hat{A}$ is the antisymmetrization operator with respect to the exchange of any pair of electrons while $F_{i\alpha}(r)$ is the radial component of the scattered electron wave function when the target is in the ith state. $\bar{\Phi}_i^\Gamma$ is called a channel function, which is obtained by coupling the target state Φ_i with the spin-angle function of the scattered electron. Let us denote the quantum numbers of the scattered electron in channel i as $k_i l_i m_{l_i} m_{s_i}$. The Hamiltonian (2.2) is diagonal with respect to the total orbital momentum L, total spin S, their projections onto the chosen axis M_L and M_S, and the parity π of the total $(N+1)$-electron system. Therefore, in the expansion (2.5) it is convenient to use the total momentum representation in which

$$\Gamma \equiv \gamma LSM_LM_S\pi \tag{2.6}$$

and the channel functions $\bar{\Phi}_i^{\Gamma}$ are defined according to the following coupling:

$$\bar{\Phi}_i^{\Gamma}\left(x_1,\ldots,x_{N+1};\hat{r}_{N+1}\sigma_{N+1}\right) = \sum_{M_{L_i}m_{l_i}}\sum_{M_{S_i}m_i}\left(L_iM_{L_i}l_im_{l_i}|LM_L\right)\left(S_iM_{S_i}\tfrac{1}{2}m_i|SM_S\right) \times \Phi_i(x_1,\ldots,x_N)Y_{l_im_{l_i}}\left(\hat{r}_{N+1}\right)\chi_{\frac{1}{2}m_i}(\sigma_{N+1}). \tag{2.7}$$

Here Y_{lm} is a spherical harmonic, $\chi(\sigma)$ is a spin function, and we use the standard notation for the Clebsch–Gordan coefficients. The function $F_{i\alpha}(x_{N+1})$ of the incident electron describes both open and closed channels. Very often the $F_{i\alpha}(x_{N+1})$ in (2.5) are called channel orbitals. If $(E-E_i)$ is positive, the channel is said to be "open"; otherwise the channel is "closed". Note, however, that the function $F_{i\alpha}(x_{N+1})$ is not quadratically integrable for open channels. Closed-channel radial functions must satisfy the same boundary conditions at $r=0$ and the same orthogonality conditions as the open channel functions, but the closed-channel functions are quadratically integrable.

The first term in expansion (2.5) should also include the integration over the continuous spectrum of the target, which corresponds to the virtual (or real) target excitation into the ionization continuum. A direct inclusion of this term in (2.5) would tremendously complicate the computational problem since the channel index becomes a continuous variable and the number of channels is not denumerable. Very often, therefore, this term is omitted. However, the continuum part of the close-coupling expansion was found to be very important at intermediate scattering energies. It can be simulated, to some extent, by the inclusion of bound pseudo-states.

The so-called correlation functions χ_i are quadratically integrable functions, usually constructed of the same one-electron orbitals as the target states Φ_i. The correlation functions ensure that the expansion (2.5) is complete in the bound orbital basis space, even though the continuum radial functions are chosen to be orthogonal to the bound orbitals as discussed below. In the central field approximation, the atomic orbitals are represented in the form

$$\varphi_j(x) = Y_{l_jm_j}(\hat{r})\chi(m_s|\sigma)\frac{1}{r}P_{n_jl_j}(r). \tag{2.8}$$

Then it is usually demanded that for $l_j = l_i$ the orthogonality condition

$$\int\limits_0^\infty P_{n_jl_j}(r)F_{i\alpha}(r)\,\mathrm{d}r = 0 \tag{2.9}$$

is satisfied. This condition does not follow from general principles and is introduced only for simplification of numerical calculations. The introduction of the correlation functions $\chi_j(x_1,\ldots,x_{N+1})$ means that in spite of implying conditions (2.9) the second sum in the expansion (2.5) permits us to take into account the possibility of the virtual capture of electrons in the unfilled subshell. The minimally necessary set of correlation functions χ_j in this case can be estimated as follows. The terms in the first sum in (2.5) can be schematically represented as Ck_jl_j, where C denotes some target state configuration. Then, after imposing the orthogonality conditions $(P_{nl}|F_{i\alpha})=0$, it is necessary to include the correlation function with the configuration Cnl, that could require recoupling of angular momenta in the case of equivalent electrons.

In the present implementation, the conditions (2.9) are optional. The use of non-orthogonal continuum orbitals allows us to avoid the introduction of the correlation functions $\chi_j(x_1,\ldots,x_{N+1})$, or to use them directly for additional inclusion of short-range correlation. In this case they also can be generated independently from the target states.

Coupled equations for the radial components of the functions $F_i(r)$ representing scattered electrons and the coefficients c_j can be obtained by substituting expansion (2.5) into the Schrödinger equation and projecting onto the target functions Φ_i and the L^2 functions χ_i. After separating out the spin and angular variables and eliminating the coefficients c_j, we obtain the following set of coupled integro-differential ("close-coupling") equations

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}r^2} - \frac{l_i(l_i+1)}{r^2} + \frac{2Z}{r} + k_i^2\right)F_i(r) = 2\sum_j(V_{ij}+W_{ij}+X_{ij})F_j(r) \tag{2.10}$$

satisfied by the radial components $F_i(r)$. In this equation, l_i is the orbital angular momentum of the scattered electron and V_{ij}, W_{ij} and X_{ij} are the partial-wave decompositions of the local direct, non-local exchange and non-local correlation potentials. These potentials are too complicated to write down explicitly except for the simplest cases. Instead they are constructed by general computer programs. Eqs. (2.10) can also contain terms which arise from the orthogonality constraints on the scattering radial functions $F_i(r)$.

A set of different computational methods for solving Eqs. (2.10) to yield the scattering matrices and amplitudes, which can be compared with experiment, has been developed over the past decades. These methods now form the basis for a number of computer program packages, some of which are widely used. As examples, we mention the linear algebraic equation method [36], the non-iterative integral equation method [37], and the convergent close-coupling method [38]. All these methods can be referred as straightforward methods where the solution of Eq. (2.10) is obtained in either configuration or momentum space for each collision energy. A promising new approach for the direct solution of the close-coupling equations (2.10), based on the B-spline basis and notable for its simplicity which is a key to a successful computational implementation, has been put forward by Froese Fischer and Idrees [39]. The method determines the required solution within a finite boundary, with no assumed boundary conditions. This is not a limitation, provided that the asymptotic region is reached, so that the solutions can be matched to a linear combination of true asymptotic solutions. The core of the algorithm involves the evaluation of the Hamiltonian and overlap matrix elements in the B-spline basis,

$$H_{ij} = \langle \Psi_i, H\Psi_j \rangle, \qquad S_{ij} = \langle \Psi_i, \Psi_j \rangle \tag{2.11}$$

and the extraction of the eigenvectors relative to the minimum modulus eigenvalues of the non-Hermitian, energy-dependent matrix

$$A(E) = H - ES \tag{2.12}$$

at each prefixed energy E. Here the very effective inverse iteration procedure is used. The B-spline approach has been further generalized to the single- and multichannel continuum case, mainly with application to photoionization of few-body system such as H, He, H^- and He^- [40–42].

2.2. The R-matrix method

The R-matrix method is one of many methods for solving the close-coupling equations (2.10). The important difference from the straightforward close-coupling formalism is a separate treatment of two regions: an inner region, in which all the electrons are fully interacting with each other and possible external fields, and an outer region, in which the continuum electron only feels a local potential. Here the coupled equations (with simple long-range potentials) are solved for each collision energy and matched, at the boundary $r = a$, to the solution in the inner region. However, instead of solving a set of coupled integro-differential equations in the internal region for each collision energy, the $(N + 1)$-electron wavefunction is expanded in terms of an energy-independent basis set and treated similarly to the N electrons in atomic bound states. Consequently, general computer codes written for bound-state atomic structure problems can be used, with only slight modifications, to generate the scattering algebra.

In the internal region, the $(N + 1)$-electron wavefunction at energy E is expanded in terms of an *energy-independent* basis set, Ψ_k, as

$$\Psi_E = \sum_k A_{Ek}\Psi_k. \tag{2.13}$$

The basis states for given total angular momenta are constructed as

$$\Psi_k^\Gamma(x_1,\dots,x_{N+1}) = \hat{A}\sum_{ij}\bar{\Phi}_i^\Gamma\bigl(x_1,\dots,x_N;\hat{r}_{N+1}\sigma_{N+1}\bigr)\frac{1}{r_{N+1}}u_j(r_{N+1})a_{ijk}^\Gamma + \sum_i \chi_i^\Gamma(x_1,\dots,x_{N+1})b_{ik}^\Gamma. \tag{2.14}$$

The u_i in Eq. (2.14) are radial continuum basis functions describing the motion of the scattering electron. They are non-zero on the boundary of the internal region and thus provide the link between the solution in the internal and external regions. The quadratically integrable functions χ_i have the same meaning as in Eq. (2.5) and are assumed to be fully confined to the internal region.

Consider now the solution of the Schrödinger equation in the internal region $[0, a]$. We note that the Hamiltonian H_{N+1} is not Hermitian in this region due to the surface terms at $r = a$ that arise from the kinetic energy operator. These surface terms can be canceled by introducing the Bloch operator

$$L_{N+1} = \sum_{i=1}^{N+1}\frac{1}{2}\delta(r_i - a)\left(\frac{\mathrm{d}}{\mathrm{d}r_i} - \frac{b-1}{r_i}\right), \tag{2.15}$$

where b is an arbitrary constant (in the present implementation we use $b = 0$). Note that $H_{N+1} + L_{N+1}$ is Hermitian for functions satisfying arbitrary boundary conditions at $r = a$. We then rewrite the Schrödinger equation in the inner region as

$$(H_{N+1} + L_{N+1} - E)\Psi = L_{N+1}\Psi. \tag{2.16}$$

This equation can be formally solved in terms of the R-matrix basis functions Ψ_k, which are obtained as eigenfunctions of the equation

$$\bigl\langle \Psi_i^\Gamma \bigl| H_{N+1} + L_{N+1} \bigr| \Psi_j^\Gamma \bigr\rangle_{\mathrm{int}} = E_i^\Gamma \bigl\langle \Psi_i^\Gamma \bigl| \Psi_j^\Gamma \bigr\rangle_{\mathrm{int}}, \tag{2.17}$$

where the integration over the radial variables is restricted to the internal region. The formal solution of Eq. (2.17) can then be expanded as

$$|\Psi^\Gamma\rangle = \sum_k |\Psi_k^\Gamma\rangle \frac{1}{E_k^\Gamma - E} \langle \Psi_k^\Gamma | L_{N+1} | \Psi^\Gamma\rangle_{\mathrm{int}}. \tag{2.18}$$

Projecting this equation onto the channel functions $\bar{\Phi}_i^\Gamma$ and evaluating on the boundary of the internal region $r_{N+1} = a$ yields

$$F_i^\Gamma(a) = \sum_j R_{ij}^\Gamma(E) \left(a \frac{\mathrm{d}F_j^\Gamma}{\mathrm{d}r} - bF_j^\Gamma \right)_{r_{N+1}=a}, \tag{2.19}$$

where we have introduced the R-matrix with elements

$$R_{ij}^\Gamma(E) = \frac{1}{2a} \sum_k \frac{w_{ik}^\Gamma w_{jk}^\Gamma}{E_k^\Gamma - E}, \tag{2.20}$$

the reduced radial wavefunctions

$$r_{N+1}^{-1} F_i^\Gamma(r_{N+1}) = \langle \bar{\Phi}_i^\Gamma | \Psi^\Gamma \rangle', \tag{2.21}$$

and the surface amplitudes

$$a^{-1} w_{ik}^\Gamma = \langle \bar{\Phi}_i^\Gamma | \Psi_k^\Gamma \rangle'_{r_{N+1}=a}. \tag{2.22}$$

The primes on the brackets in Eqs. (2.21) and (2.22) mean that the integration is carried out over all the electronic space- and spin-coordinates except for the radial coordinate r_{N+1} of the scattered electron. Eqs. (2.19) and (2.20) are the basic equations describing the scattering of electrons by atoms or ions in the internal region. Together with the following relations for the coefficients A_{Ek} in Eq. (2.13),

$$A_{Ek} = \frac{1}{2a}(E_k - E)^{-1} \sum_i w_{ik}(a) \left(a \frac{\mathrm{d}F_i}{\mathrm{d}r} - bF_i \right)_{r=a} = \frac{1}{2a}(E_k - E)^{-1} \boldsymbol{w}^{\mathrm{T}} \boldsymbol{R}^{-1} \boldsymbol{F}, \tag{2.23}$$

they allow us to establish the total wavefunction Ψ_E in the inner region for any value of the total energy E given the values of the scattering orbitals on the boundary. The R-matrix given by Eq. (2.20) is obtained at all energies by diagonalizing $H_{N+1} + L_{N+1}$ once for each set of conserved quantum numbers Γ to determine the basis functions Ψ_k and the corresponding eigenenergies E_k. The logarithmic derivatives of the continuum radial wavefunctions $F_i(r)$ on the boundary of the internal region are then given by Eq. (2.19).

An important point in the R-matrix method is the choice of the radial continuum basis functions u_j in Eq. (2.14). Although members of any complete set of functions satisfying arbitrary boundary conditions can be used, a careful choice will speed up the convergence of the expansion (2.13). In the standard R-matrix approach developed by the Belfast group [4] numerical basis functions satisfying homogeneous boundary conditions at $r = a$ were adopted. This approach gives accurate results, provided that corrections proposed by Buttle [16] to allow for the omitted high-lying poles in the R-matrix expansion are included. The shortcoming of this approach is that all basis functions have the same (usually, zero) logarithmic derivative at the boundary of the internal region. This yields a discontinuity in the slope of the resulting continuum orbitals. In the standard approach, the basis functions u_i are forced to be orthogonal to the bound orbitals P_{nl} used for the construction of the target wavefunctions. To compensate for the resulting restrictions on the total wavefunctions, the basis states Ψ_k must contain the correlation functions χ_i as in Eq. (2.14). In the case of complex atoms, when extensive many-configuration expansions are used for accurate representations of the target state wave functions, this may lead to a very large number of the correlation functions χ_i which must be included in the close-coupling expansion to compensate for the orthogonality constraints. The present program has the option to imply a fully non-orthogonal set of basis functions u_j in Eq. (2.14). It allows us to use the close-coupling expansions (2.14) without any correlation functions at all. Nevertheless, in practical calculations some restricted orthogonality constraints are usually imposed on the basis functions u_j, depending on the physical model under consideration.

The other key point of the present approach is the use of B-splines instead of the one-electron basis functions $u_i(r)$ in the R-matrix representation (2.14) of the inner region. Full description of B-splines is given in Section 12. The B-splines possess properties as though they were especially created for the R-matrix theory. They form a complete basis on the finite interval $[0, R]$, have a universal nature, and are very convenient in numerical calculations because they allow us to avoid finite-difference formulae. Here we have to distinguish between using B-splines as another basis for representation of the one-electron orbitals and using B-splines for generating the complete pseudo-spectrum for some one-electron Hamiltonian, as is done in many atomic structure calculations [26]. The present program provides both options. In the first case, the coefficients a_{ijk} are found from the diagonalization (2.17) of the full Hamiltonian in the B-spline representation. Such an approach is more suited for bound-state calculations. In the second case, we first perform a preliminary diagonalization of the Hamiltonian blocks corresponding to one channel. This generates a complete set of one-electron orbitals for each channel. We can transform the Hamiltonian matrix to the new representation based on these one-electron orbitals, and now we are able to reduce the dimension of the full interaction matrix by dropping some of the basis orbitals, depending on the problem under consideration.

The boundary conditions in the B-spline basis define only the first and the last basis functions, which are the only non-zero terms, respectively, for $r = 0$ and $r = a$. The boundary conditions for the scattering function at the origin are satisfied in the form $F(0) = 0$ by simply removing the first B-spline from the basis set. For the definition of the R-matrix at the boundary (2.20), the amplitudes of the wavefunctions at $r = a$, w_{ki}, are required. These values are defined by the coefficients of the last spline, which is equal to one at the boundary. The summation over the entire expansion (2.13) now gives the surface amplitudes in a straightforward manner. Details for using B-splines in the construction and diagonalization of the interaction matrix (2.17) are given in Sections 7 and 8, respectively, along with a description of the corresponding programs.

2.3. The external region

The next step in the calculation is to solve the scattering problem in the external region and to match the solutions on the boundary $r = a$ in order to obtain the K-matrices, S-matrices, or phase shifts. Since the radius a is chosen such that electron exchange is negligible in this region, we can expand the total wavefunction in the form

$$\Psi^{\Gamma}(x_1,\dots,x_{N+1}) = \sum_j \bar{\Phi}_i^{\Gamma}\big(x_1,\dots,x_N;\hat{r}_{N+1}\sigma_{N+1}\big) r_{N+1}^{-1} F_i^{\Gamma}(r_{N+1}), \quad r_{N+1} > a, \tag{2.24}$$

where the $\bar{\Phi}_i^{\Gamma}$ are the same set of channel functions as those retained in expansion (2.5) and the $F_i(r)$ are the analytic continuations for $r > a$ of the reduced radial wavefunctions defined by Eqs. (2.21). The radial functions $F_i(r)$ satisfy the set of coupled differential equations

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}r^2} - \frac{l_i(l_i+1)}{r^2} + \frac{2(Z-N)}{r} + k_i^2\right) F_i^{\Gamma}(r) = 2\sum_{j=1}^{n}\sum_{\lambda=1}^{\Lambda} \frac{a_{ij}^{\lambda}}{r^{\lambda+1}} F_j^{\Gamma}(r), \quad i = 1, n \ (r \geqslant a), \tag{2.25}$$

where n is the number of channel functions retained in expansions (2.5) and (2.24) while l_i and k_i^2 are the channel angular momenta and energies. The interaction between channels is defined by the long-range potential with coefficients

$$a_{ij}^{\lambda,\Gamma} = \Big\langle \bar{\Phi}_i^{\Gamma}\big(x_1,\dots,x_N;\hat{r}_{N+1}\sigma_{N+1}\big)\Big|\sum_{k=1}^{N} r_k^{\lambda} P_{\lambda}(\cos\theta_{kN+1})\Big|\bar{\Phi}_j^{\Gamma}\big(x_1,\dots,x_N;\hat{r}_{N+1}\sigma_{N+1}\big)\Big\rangle, \tag{2.26}$$

where $\cos\theta_{kN+1} = \hat{r}_k \,.\, \hat{r}_{N+1}$, $P_{\lambda}(x)$ is a Legendre polynomial. The integral in Eq. (2.26) is carried out over all electronic space and spin coordinates except for the radial coordinate of the scattered electron. In practical calculations, the long-range potential coefficients are the by-product of generating the interacting matrix (2.17) and are defined by the coefficients of the relevant Slater integrals R^k, describing the direct interaction between channels in the internal region. Note that $a_{ij}^0 = N\delta_{ij}$ because of the orthogonality of the target wavefunctions Φ_i. This relation is used to check the target representation in the present program.

Eqs. (2.25) can be integrated outward from $r = a$ and fitted to an asymptotic expansion at large r as described in [28]. If all n scattering channels are open, the asymptotic form of the radial wavefunctions $F_i(r)$ may be written in the form

$$\boldsymbol{F}(r) \underset{r\to\infty}{\sim} \boldsymbol{k}^{-1/2}(\boldsymbol{F} + \boldsymbol{G}\boldsymbol{K}), \tag{2.27}$$

where we have written the channel momenta, $\boldsymbol{k}$, as a diagonal matrix. The diagonal matrices $\boldsymbol{F}$ and $\boldsymbol{G}$ correspond to regular and irregular Coulomb (or Riccatti–Bessel) functions in each scattering channel. The asymptotic expression (2.27) defines the reaction K-matrix, $\boldsymbol{K}$, appropriate for standing-wave boundary conditions. The K-matrix is related to the scattering S-matrix by the equation

$$\boldsymbol{S} = \frac{1 + i\boldsymbol{K}}{1 - i\boldsymbol{K}}. \tag{2.28}$$

It may be used directly to calculate cross sections and other scattering observables.

The long-range coefficients α_{ij} together with the target energies and the definition of the structure of the close-coupling equations constitute the information needed to solve the scattering problem in the external region. This problem is well developed, and there is a set of general computer codes for its solution. An efficient code is the outer region program FARM [28], which uses the R-matrix propogator technique. In the present implementation, we use the program ASYPCK [43] for treating the external region. Note that, in addition to the K-matrix, we also need the outer region solutions at the boundary $r = a$ for the calculation of photoionization cross sections.

2.4. Radiative processes

In R-matrix theory, the photoionization cross-sections can be defined through the dipole matrix between the initial state Φ_0 and the R-matrix basis states Ψ_k, provided that all radial orbitals of the initial state are well confined to the inner region. The total photoionization cross section for a given photon energy ω is

$$\sigma(\omega) = \left(\frac{4}{3}\pi^2 a_0^2 \alpha\right)\left(\frac{\omega C}{2L_0+1}\right)\sum_j \left|(\Psi_j^- \| D \| \Phi_0)\right|^2, \tag{2.29}$$

where $\boldsymbol{D}$ is the electric dipole operator. It could be either in the length or the velocity form, with $C = 1$ in the length form, and $C = 4/\omega^2$ in the velocity form, and the photon energy being in Ry. The index j runs over different possible solutions, and the other quantities have their usual meaning. The solutions Ψ_j^- in (2.29) correspond to asymptotic conditions with a plane wave in the direction of the ejected electron momentum $\boldsymbol{k}$ and ingoing waves in all open channels. The corresponding radial functions $\boldsymbol{F}^-$are related to the $\boldsymbol{F}(r)$ with the K-matrix asymptotic form (2.27) via

$$\boldsymbol{F}^- = -i\boldsymbol{F}(1 - i\boldsymbol{K})^{-1}. \tag{2.30}$$

Expanding Ψ_j^- in terms of the R-matrix states as in Eq. (2.13) and using the expressions (2.23), we find that

$$(\Psi_j^- \| D \| \Phi_0) = \frac{1}{a}\sum_k \frac{(\Psi_k \| D \| \Phi_0)}{E_k - E_0 - \omega}\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{F}_j^-(a) \tag{2.31}$$

where $(\Psi_k \| D \| \Phi_0)$ are reduced matrix elements between the initial state and the R-matrix basis functions.

In addition to the total cross section, another important quantity is the anisotropy parameter β, which defines the angular distribution of photoelectrons. For example, for linearly polarized incident radiation the angular distribution of photoelectrons is given by

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\hat{k}} = \frac{\sigma}{4\pi}\left[1 + \beta P_2(\cos\theta)\right],$$

where θ is the angle of the ejected electron relative to the axis of polarization, while for unpolarized radiation it takes the form

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\hat{k}} = \frac{\sigma}{4\pi}\left[1 - \frac{\beta}{2}P_2(\cos\theta)\right],$$

where θ is the angle of the ejected electron relative to the incident radiation beam. β is determined from [2]:

$$\beta = \frac{15(-1)^{L_0+L_f}}{\sum_j |(\Psi_j^- \| D \| \Phi_0)|^2}\sum_{jj'}\left[(2l+1)(2l'+1)(2L+1)(2L'+1)\right]^{1/2}$$
$$\times\, i^{l-l'}\mathrm{e}^{-i\sigma_l + i\sigma_{l'}}\begin{Bmatrix} L & l & L_f \\ l' & L' & 2 \end{Bmatrix}\begin{Bmatrix} L & 1 & L_0 \\ 1 & L' & 2 \end{Bmatrix}\begin{pmatrix} l & l' & 2 \\ 0 & 0 & 0 \end{pmatrix}(\Phi_0 \| D \| \Psi_{j'}^-)(\Psi_j^- \| D \| \Phi_0),$$

where L_o and L_f are the angular momenta of the initial atomic state and final target states, respectively, and the indexes j and j' stand for the continuum channels, associated with the given target state. σ_l stands for Coulomb phase, see (2.34). Here and below the standard notation for the nj symbols is used.

In order to use the expression (2.31), we need the values of the solutions $F_i(a)$ at the R-matrix boundary. They can be obtained by matching the general asymptotic solutions of Eq. (2.25) to the solutions in the internal region at $r = a$. This can be done with relation (2.19), which in matrix form reads

$$\boldsymbol{F} = a\boldsymbol{R}\boldsymbol{F}' - b\boldsymbol{R}\boldsymbol{F} \quad (r \leqslant a). \tag{2.32}$$

Note that in the outer region we have n_o independent physical solutions, where n_o is the number of open channels, which is defined by all the target states accessible at a given excitation energy. To relate the $n \times n$-dimensional R-matrix to the $n_o \times n_o$ K-matrix defined in Eq. (2.27), we introduce $n + n_o$ linearly independent solutions $s_{ij}(r)$ and $c_{ij}(r)$ of Eq. (2.25) satisfying the boundary conditions

$$\left.\begin{matrix} s_{ij}(r) \\ c_{ij}(r) \\ c_{ij}(r) \end{matrix}\right\} \underset{r\to\infty}{\sim} \begin{cases} \sin\theta_i\delta_{ij}, & i = 1, n,\ j = 1, n_o, \\ \cos\theta_i\delta_{ij}, & i = 1, n,\ j = 1, n_o, \\ \exp(-\phi_i)\delta_{i,j-n_o}, & i = 1, n,\ j = n_o + 1, n, \end{cases} \tag{2.33}$$

where θ_i and ϕ_i define the asymptotic phases in the open and closed channels, respectively:

$$\begin{aligned}\theta_i &= k_i r - \frac{1}{2}l_i\pi - \frac{z}{k_i}\ln 2k_i r + \arg\Gamma\left(l_i + 1 + i\frac{z}{k_i}\right),\\ \phi_i &= |k_i|r - \frac{z}{|k_i|}\ln(2|k_i|r).\end{aligned} \tag{2.34}$$

Now we can rewrite the asymptotic form of the scattering wavefunction in the general form

$$\boldsymbol{F} = \boldsymbol{s} + \boldsymbol{c}\boldsymbol{K} \quad (r \geqslant a). \tag{2.35}$$

Substituting this into Eq. (2.32) and solving for $\boldsymbol{K}$, we get

$$\boldsymbol{K} = \boldsymbol{B}^{-1}\boldsymbol{A}, \tag{2.36}$$

where

$$\boldsymbol{A} = -\boldsymbol{s} + a\boldsymbol{R}\left(\boldsymbol{s}' - \frac{b}{a}\boldsymbol{s}\right), \qquad \boldsymbol{B} = +\boldsymbol{c} - a\boldsymbol{R}\left(\boldsymbol{c}' - \frac{b}{a}\boldsymbol{c}\right). \tag{2.37}$$

This completes the evaluation of the reactance matrix $\boldsymbol{K}$ and the value $\boldsymbol{F}(a)$, provided that the asymptotic solutions $s_{ij}(r)$ and $c_{ij}(r)$ are known. As mentioned above, a number of computer packages are available for obtaining these solutions, such as the program ASYPCK [43], which we use in the present package.

The initial state Φ_0 in the present program can be obtained either in an independent MCHF calculation, or in the framework of B-spline bound-states calculations discussed in the next section. In general, it requires the evaluation of dipole matrix elements between states with non-orthogonal orbitals. Details of the calculation of dipole matrix elements in this case are presented in Sections 9 and 10.

2.5. Bound-state calculations

Electron–atom collision theory is concerned with states of an $(N+1)$-electron system for which N electrons are bound in an atom or atomic ion and one electron can escape to infinity. Such states may be represented accurately using the close-coupling expansion (2.5). This expansion is also suitable for representing states with one electron highly excited and the other N electrons more tightly bound. When the close-coupling approximation is used to calculate bound states of atomic systems, it is referred to as the frozen-core (FCS) approximation, and the Φ_i are usually labeled ‘core’ rather than ‘target’ functions.

The FCS method has several advantages. It can readily be extended to highly excited states. The multichannel form of Eq. (2.5) allows us to include explicitly the interaction between different Rydberg series, as well as the interaction of the Rydberg series with perturbers that can be represented in the second part of the expansion. The energies and wavefunctions can be computed efficiently with an accuracy comparable to that obtainable using the best alternative methods, from which a large amount of radiative data can be generated. The same expansions can be used for close-coupling collision calculations as for FCS bound-state calculations. Often a comparison of the calculated bound-state energies with experimental energies provides a check on the accuracy of the collision calculations. At the same time, one can take advantage of the extensive experience accumulated from close-coupling collision calculations and the codes developed for such calculations, and one can apply them to the study of Rydberg series. The first extensive FCS calculations for many-electron systems were made using the computer program IMPACT [36] that solved the resulting integro-differential equations (see, for example, [44–46]). Further developments of the FCS method for the study of Rydberg series were made in connection with the R-matrix method [47]. An example of such a calculation is given by Berrington and Seaton [48].

The accuracy of both the above-mentioned methods is restricted through the use of finite-difference methods for computing the radial functions. More accurate numerical results can be obtained by using a spline FCS method for Rydberg series, in which the wavefunctions of the outer electrons are expanded directly in B-splines in some finite region $r \leqslant a$, with a sufficiently large value of a. Such a method is implemented in the present package. The zero boundary conditions are imposed by deleting from the expansion the first and the last splines, i.e. the only splines with a nonzero value at the boundaries. We also delete the next to last spline, in order to guarantee a zero derivative at the border for all bound solutions.

The choice of B-splines as basis functions has some advantages. The completeness of the B-spline basis ensures that, in principle, we can study the entire Rydberg series. The number of physical states we can obtain in one diagonalization is defined by the box radius a, which can easily be varied in the B-spline representation. Exponential grid was found to be quite suitable for bound states, and it allows us to use a rather large radius with a relatively small number of B-splines. For example, in order to obtain Rydberg states up to $n = 10$ in neutral atoms, it is often sufficient to choose the box radius equal to 300 a.u. and the number of splines as 45. If we aim to study Rydberg series up to $n = 20$, we should increase the border radius to 1200 a.u. With an exponential grid, this increases the number of splines only to 51. Hence, the size of the interaction matrix, which is proportional to the number of splines, does not increase considerably. Of course, these numbers somewhat depend on the nuclear charge Z. For very high Z, it is advisable to add a few splines at very small radii to achieve an accurate representation of the orbitals near the nucleus.

The wavefunctions in this method are obtained for all radii and for all Rydberg states under consideration. There is no need to obtain an asymptotic solution and to match it to the inner-region solution as in the R-matrix method discussed above. This considerably simplifies the calculations and the codes, but the use of a finite B-spline basis limits the upper n-value for the Rydberg states. Consequently, this method can be efficient for the study of moderately excited states with values of n not too large, typically in the range 10–30.

Our implementation of the spline method differs from a previous one [6] through the use of non-orthogonal orbitals, both for the construction of target wavefunctions and for the representation of the outer electron. It provides us with a great deal of flexibility in the choice of the 'target' wavefunctions, which can be optimized for each atomic state separately, and in the introduction of different correlation corrections. In the present program, the core–core correlation may be taken into account by using extensive multi-configuration target states. The core-valence correlation can be introduced, in principle, in two ways, either by using a large set of excited target states in the close-coupling expansion or by introducing additional $(N+1)$-electron states, specially designed for this purpose. The convergence of the close-coupling expansion can be very slow, and hence the first approach is much more time consuming. Nevertheless, our experience shows that this method gives a more accurate description of the core-polarization potential, and the $(N+1)$-electron terms in Eq. (2.5) are better used only for the inclusion of the short-range correlation.

Another difference compared to earlier implementations of B-splines concerns the relativistic corrections. Usually (see, e.g., [49,50]) relativistic corrections are introduced via the Breit–Pauli CI approach after having calculated nonrelativistic wavefunctions using the B-spline expansions for the relevant states. We include the Breit–Pauli operators directly into the secular equations (2.17). Hence, our B-spline expansions for Rydberg electrons include both the relativistic and correlation corrections.

2.6. Relativistic corrections

With increasing nuclear charge Z, relativistic effects become important both in the target description and the scattering wavefunctions, even for low-energy electron scattering. In the current code, relativistic corrections can be included through the Breit–Pauli Hamiltonian.

The Breit–Pauli Hamiltonian [51] can be considered as a first-order correction to the non-relativistic atomic Hamiltonian. It includes all relativistic terms up to order $(\alpha Z)^2$, where α is the fine-structure constant. According to their different angular symmetry and different influence on the atomic structure, we recognize three parts of the Breit–Pauli Hamiltonian, namely

$$H_{\mathrm{BP}} = H_{\mathrm{NR}} + H_{\mathrm{RS}} + H_{\mathrm{FS}}. \tag{2.38}$$

Here H_{NR} is the ordinary non-relativistic many-electron Hamiltonian

$$H_{\mathrm{NR}} = -\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - Z\sum_{i=1}^{N}\frac{1}{r_i} + \sum_{i<j}\frac{1}{r_{ij}}. \tag{2.39}$$

The *relativistic shift* operator H_{RS} commutes with $\boldsymbol{L}$ and $\boldsymbol{S}$ and can be written as

$$H_{\mathrm{RS}} = H_{\mathrm{MC}} + H_{\mathrm{D1}} + H_{\mathrm{D2}} + H_{\mathrm{OO}} + H_{\mathrm{SSC}}, \tag{2.40}$$

where H_{MC} is the *mass correction* term

$$H_{\mathrm{MC}} = -\frac{\alpha^2}{8}\sum_{i=1}^{N}(\nabla_i^2)^+ \cdot \nabla_i^2, \tag{2.41}$$

while H_{D1} and H_{D2} are the one-body and two-body Darwin terms, i.e. the relativistic correction to the potential energy,

$$H_{\mathrm{D1}} = -\frac{\alpha^2 Z}{8}\sum_{i=1}^{N}\nabla_i^2\left(\frac{1}{r_i}\right), \tag{2.42}$$

$$H_{\mathrm{D2}} = \frac{\alpha^2}{4}\sum_{i<j}\nabla_i^2\left(\frac{1}{r_{ij}}\right). \tag{2.43}$$

Next, H_{SSC} is the *spin–spin contact* term

$$H_{\mathrm{SSC}} = -\frac{8\pi\alpha^2}{3}\sum_{i<j}^{N}(\boldsymbol{s}_i\cdot\boldsymbol{s}_j)\delta(\boldsymbol{r}_i\cdot\boldsymbol{r}_j) \tag{2.44}$$

and, finally, H_{OO} is the *orbit–orbit* term

$$H_{\mathrm{OO}} = -\frac{\alpha^2}{2}\sum_{i<j}^{N}\frac{1}{r_{ij}^3}\left\{\frac{(\boldsymbol{p}_i\cdot\boldsymbol{p}_j)}{r_{ij}} + \frac{\boldsymbol{r}_{ij}(\boldsymbol{r}_{ij}\cdot\boldsymbol{p}_i)\cdot\boldsymbol{p}_j}{r_{ij}^3}\right\}. \tag{2.45}$$

The *fine-structure* operator H_{FS} describes interactions between the spin and the orbital angular momenta of the electrons. It does not commute with $\boldsymbol{L}$ and $\boldsymbol{S}$ but only with the total electronic angular momentum $\boldsymbol{J} = \boldsymbol{L} + \boldsymbol{S}$. The fine-structure operator consists of three terms,

$$H_{\mathrm{FS}} = H_{\mathrm{SO}} + H_{\mathrm{SOO}} + H_{\mathrm{SS}}. \tag{2.46}$$

Here H_{SO} is the *nuclear spin–orbit* term

$$H_{\mathrm{SO}} = \frac{\alpha^2 Z}{2}\sum_{i=1}^{N}\frac{1}{r_i^3}(\boldsymbol{l}_i\cdot\boldsymbol{s}_i), \tag{2.47}$$

H_{SOO} is the *spin-other-orbit* term

$$H_{\mathrm{SOO}} = -\frac{\alpha^2}{2}\sum_{i\neq j}^{N}\frac{\boldsymbol{r}_{ij}\times\boldsymbol{p}_i}{r_{ij}^3}(\boldsymbol{s}_i + 2\boldsymbol{s}_j), \tag{2.48}$$

and H_{SS} is the *spin–spin* term

$$H_{\mathrm{SS}} = \alpha^2\sum_{i<j}^{N}\frac{1}{r_{ij}^3}\left\{(\boldsymbol{s}_i\cdot\boldsymbol{s}_j) - 3\frac{(\boldsymbol{s}_i\cdot\boldsymbol{r}_{ij})(\boldsymbol{s}_j\cdot\boldsymbol{r}_{ij})}{r_{ij}^2}\right\}. \tag{2.49}$$

The present code has the option of directly including all the Breit–Pauli terms in the interaction matrix. For most applications, however, it is sufficient to only include the one-electron terms and the two-electron spin-other-orbit interaction. In contrast to the Belfast R-matrix code, the present code does not use recoupling to include the relativistic terms. Hence, storage and time problems associated with a possibly very large number of non-relativistic Hamiltonian matrices to be calculated first can be avoided. An exception is made for the mass correction term (2.41). Direct inclusion of this term in the interaction matrix often leads to non-physical R-matrix poles with very low energies. For this reason, the program has the option to include this term as a first-order perturbation, as described further in Section 8.

The non-fine-structure interactions commute with $\boldsymbol{S}^2$, $\boldsymbol{S}_{\mathrm{z}}$, $\boldsymbol{L}^2$ and $\boldsymbol{L}_{\mathrm{z}}$ and can therefore be considered in the $|\beta LSM_LM_S\rangle$ representation. The fine-structure interaction only commutes with $\boldsymbol{S}^2$, $\boldsymbol{L}^2$, $\boldsymbol{J}^2$ and $\boldsymbol{J}_{\mathrm{z}}$, where $\boldsymbol{J}$ and $\boldsymbol{J}_{\mathrm{z}}$ are the total electronic angular momentum and its projection on the quantization axis, respectively. Here we must use the $|\gamma LSJM_J\rangle$ representation for the total configuration state functions

$$\Phi(\gamma LSJM_J) = \sum_{M_LM_S}(LM_LSM_S|JM_J)\Phi(\gamma LSM_LM_S). \tag{2.50}$$

When we include the fine-structure interactions, we must define the pair-coupling scheme in the close-coupling expansion (2.5). The present program has three options for the definition of the pair-coupling scheme. The first option is *LSJ*-coupling. It is applied only in the bound-state calculations as

$$\boldsymbol{L}_i + \boldsymbol{l} = \boldsymbol{L}, \qquad \boldsymbol{S}_i + \tfrac{1}{2} = \boldsymbol{S} \quad \text{and} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{J}. \tag{2.51}$$

Here L_i and S_i define the target term while l and $\frac{1}{2}$ are the orbital angular momentum and the spin of the valence electron. This scheme allows us to use simpler *LS* expansions for the target states and also reduces the number of channels (Rydberg series) in the FCS expansion.

The most popular scheme for scattering calculations is the *jK* pair coupling

$$\boldsymbol{J}_i + \boldsymbol{l} = \boldsymbol{K} \quad \text{and} \quad \boldsymbol{K} + \tfrac{1}{2} = \boldsymbol{J}, \tag{2.52}$$

where J_i is the total electronic angular momentum of the target state. The target expansions should now be specified for each fine-structure level. The corresponding Hamiltonian matrix (2.17) then becomes much larger. This results in more coupled channels in the close-coupling expansion (2.5), therefore requiring a considerable increase in the computational effort.

The last coupling option is the jj pair coupling scheme

$$\boldsymbol{l} + \tfrac{1}{2} = \boldsymbol{j} \quad \text{and} \quad \boldsymbol{J}_i + \boldsymbol{j} = \boldsymbol{J}, \tag{2.53}$$

which can be useful in case of a strong spin–orbit interaction.

3. General features of the BSR 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 low-energy electron/photon scattering from neutral atoms, where we believe 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 atomic target states. The most popular ones are the MCHF Atomic Structure Package by Froese Fischer et al. [52], the CIV3 program by Hibbert [53], and SUPERSTRUCTURE [54], but others can be used as well. All these programs use different formats for the presentation of the one-electron orbitals and the configuration expansions. As a basic representation for the different atomic states we choose the format of the MCHF package. Here the configuration expansion for a 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 *bsw*-files). These files can be generated from the w-files in the MCHF package, or from the Slater-type representation used in CIV3, by using the interface programs described in Section 13. 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 Belfast 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 to 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.

3.2. *Main programs and data files*

The program modules are run sequentially as outlined in Fig. 1. Besides a set of c- and *bsw*-files for the target description, the user should prepare three input files (**knot.dat, target, bsr_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 12.2). **target** contains the description of all target states and partial waves for the given run (see program BSR_PREP and BSR_CONF for details). **bsr_par** defines the input parameters which control the execution of the programs. All input parameters in **bsr_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 **bsr_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. 1 in bold, are produced by the 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 the BSR_HD and BSR_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 can then choose the program for the treatment of the external region. In the case of photoionization calculations, two additional programs, 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 [43]. 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 amount 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.

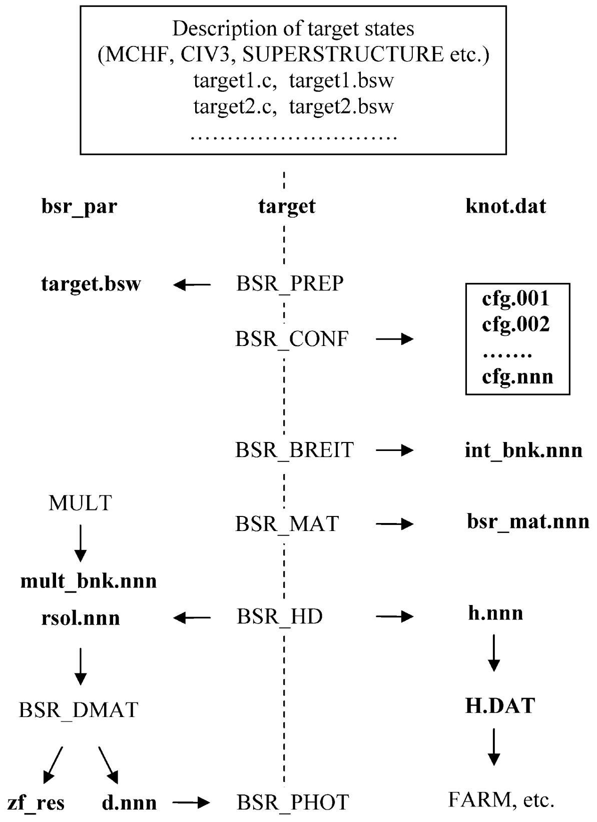

Fig. 1. Structure of the BSR package. Notation: capital names stand for the programs; the vertical dashed line indicates the sequential execution of the programs, from top to bottom; bold names stand for the most important intermediate data files.

All unit numbers and file names are defined within the program. They are collected in the corresponding modules *param*, along with all other parameters used 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 BSR 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 BSR package is organized as a set of main programs, shown in Fig. 1, utility programs, which help the user to prepare and process the input/output data, and four libraries BSPLINE, ZCONF, ZCOM and LAPACK, which contain routines common to all programs. The BSPLINE library, one of the most important parts of the package, contains routines for using B-splines in the atomic structure calculations. The ZCONF 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 for the linear-algebra calculations, such as matrix diagonalization or solving a system of linear equations. For the compilation of the BSR program, a UNIX/LINUX user can use the *make*-files provided along with the source files. Other users can use these files as a guide for compilation in a specific environment.

4. Program BSR_PREP

4.1. Outline of the BSR_PREP calculations

This program provides initial preparations for BSR calculations. Since the input target states may be generated in independent calculations, it is possible that the same spectroscopic notation was given to the different orbitals. For example, we can use different term-dependent radial functions for the same *nl*-orbital, or simply use the same spectroscopic notation for the correlated functions in different states. In order to distinguish between such orbitals, we use an additional ‘set number’ or ‘set index’, e.g., $nl_1, nl_2, \dots, nl_a, nl_b$ and so on. It is difficult and very inconvenient for the user to keep track of all such cases by hand. The

BSR_PREP program therefore analyses the input target files and, if necessary, assigns an individual set number to each orbital. The same concerns also the additional sets of $(N+1)$-electron configurations included in the close-coupling expansion (2.14) for a more complete description of short-range correlations. We will call these configurations 'perturbers', because they are often used in bound-state calculations to describe the states that perturb the Rydberg series under consideration. These additional $(N+1)$-electron configurations should be provided in separate files, and they should be distinguished from the configurations which are automatically generated for compensation of the orthogonality constraints imposed on the continuum orbitals (see the description of the BSR_CONF program).

The list of input target states and perturbers are provided by the user in the file **target**. BSR_PREP sequentially reads all *c*- and *bsw*-files and compares the one-electron radial functions. They are considered the same if their overlaps or first moments differ by less than a given tolerance:

$$\begin{aligned} &\bigl|\langle P_1|P_2\rangle - 1\bigr| \leqslant \varepsilon, \\ &\bigl|\langle P_1|r|P_1\rangle - \langle P_2|r|P_2\rangle\bigr| \leqslant \varepsilon, \\ &\bigl|\langle P_1|1/r|P_1\rangle - \langle P_2|1/r|P_2\rangle\bigr| \leqslant \varepsilon. \end{aligned} \tag{4.1}$$

The radial functions are considered orthogonal if their overlap

$$\bigl|\langle P_1|P_2\rangle\bigr| \leqslant \varepsilon. \tag{4.2}$$

For a given subset of orthogonal orbitals, BSR_PREP assigns the same set number. There are practically no restrictions on the number of different non-orthogonal orbitals.

Then BSR_PREP rewrites the target *c*- and *bsw*-files with a new consistent spectroscopic notation. New target *c*-files receive the names **targ_001.c**, **targ_002.c**, and so on, and they are ordered according to the target energies. All target radial functions are placed in the single file **target.bsw**. The same procedure is then applied to the perturber files, and each perturber is now rewritten in the pair of files **pert_nnn.c** and **pert_nnn.bsw**, where **nnn** denotes the corresponding partial wave. All the above rewritings are recorded in the file **target**, from which the user can obtain the relation between the new and the input names.

The only input parameter for BSR_PREP, which can be provided in the command line or in the input file **bsr_par**, is **`eps_ovr`** – the value of the tolerance in Eqs. (4.1), (4.2). Its default value of 10^{-7} is usually sufficient for most applications.

4.2. Data files

target File type: formatted sequential input.
Written by user.
Read and modified by program BSR_PREP and BSR_CONF.
Description: list of target states and scattering channels.
Format: see Section 4.3.

bsr_par File type: formatted sequential input.
Written by user.
Description: input parameters for given run.
Format: see Section 3.2 and examples in Section 14.

knot.dat File type: formatted sequential input.
Written by user.
Read and modified by routine **define_grid** from BSPLINE library.
Description: input parameters that define the B-spline grid.
Format: see Section 12.2.

name.c File type: formatted sequential input.
Provided by user.
Read by routine **r_confc** from ZCONF library.
Description: configuration expansion for one target state.
Format: see Section 12.5.

name.bsw File type: unformatted sequential input.
Created by utilities **w_bsw** or **slater_bsw** (see Section 13).
Read by routine **read_bsw**.
Description: contains B-spline representation of one-electron radial functions.

targ_nnn.c	File type: formatted sequential output. Created by program BSR_PREP. Read by program BSR_CONF. Description: contains configuration expansion for one target state **nnn**, with consistent spectroscopic notation for one-electron orbitals.
target.bsw	File type: unformatted sequential input. Created by program BSR_PREP. Read by program BSR_MAT. Description: contains all one-electron target orbitals in the B-spline basis.
pert_nnn.c	File type: formatted sequential output. Created by program BSR_PREP. Read by program BSR_CONF. Description: contains configuration expansion for perturber **nnn**, with consistent spectroscopic notation for one-electron orbitals.
pert_nnn.bsw	File type: unformatted sequential output. Created by program BSR_PREP. Read by programs BSR_MAT. Description: contains one-electron orbitals in B-spline basis for perturber **nnn**.
bsr_prep.log	File type: formatted sequential output. Written by program BSR_PREP. Read by user. Description: running information.

4.3. *Input data in file **target***

This file is created by the user and contains the description of the physical scattering model under consideration. An example of an input **target** file is given in Fig. 2. In addition to the data, the file also contains dashed lines and comments placed behind the '!' character. The comments are optional, whereas the dashed lines are used to delimit different types of data and are part of the format. The first line is a title for the given run. Then, after a line-delimiter, the user provides some basic data such as the type of coupling (*LS* or semi-relativistic *JK*, *JJ*), the number of electrons, and the atomic number. The format for all quantities is the same as for the parameters in the file **bsr_par: `name` = `value`**, where name should be placed at the beginning of the line. The next block of data is the list of target c-files, which should be placed after the number of target states, **`ntarg`**. Then the user provides the list of partial waves, placed after the number of partial waves, **`nlsp`**. The description of each partial wave contains the total term $(L, 2S+1, \pi)$ in *LS* coupling or $(2J, 0, \pi)$ in semi-relativistic calculations. The perturber for a given partial wave, if any, is given by the name of the corresponding c-file, otherwise the user should indicate **`no`**.

BSR_PREP adds some additional information to the **target** file. An example of **target** after a run of BSR_PREP is given in Fig. 3. Now the target states are ordered by energy, and for each target state the program provides a new name **targ_nnn.c**, the term $(L, 2S+1, \pi)$, the energy, the number of configurations, and the number of new one-electron radial orbitals used in the description of a given target state. All this information was extracted from the corresponding target c-files. Similar information is also provided for the perturbers: a new standard name **pert_nnn.c**, the number of configurations and the number of new one-electron radial orbitals used in the description of a given perturber. Also provided are the total number of target configurations, **`nct`**, the total number of one-electron orbitals, **`nwt`**, and the list of orbitals with new spectroscopic notation, including the set numbers. In the example shown, we used two different sets of orthogonal orbitals, for even-parity and odd-parity states, respectively. These sets differ by their indexes 1 and 2. Note that the 1s orbital represents the closed shell 1s^2, and it is common for all configurations. These core orbitals should be orthogonal to all orbitals, and hence they do not need any additional set index.

5. Program BSR_CONF

5.1. *Outline of the BSR_CONF calculations*

This program generates the close-coupling expansions (2.5) for the case under consideration, based on the information given in the file **target**. First, the program reads the target configuration expansions from all **targ_nnn.c** files. For each partial wave, the corresponding close-coupling expansions are then generated in *LS*-coupling, or in *LSJ*-, *jK*- or *jj*-coupling in the case of semi-relativistic calculations, according to the input parameter **coupling**. The continuum configurations are simply generated from the target configurations by adding the continuum orbital, in accordance with the angular momentum coupling rules. All possible

```
 e + C
---------------------------------------------------------------
coupling = LS   !   non-relativistic calculations
nz = 6          !   nuclear charge
nelc = 6        !   number of electron
---------------------------------------------------------------
ntarg = 36      !   number of target states
---------------------------------------------------------------
2p2_1S.c
2p2_1D.c
2p2_3P.c
2p3_3Do.c
2p3_5S.c
2p3s_3Po.c
2p3s_1Po.c
2p3p_1P.c
..........
3Po_ps4.c
3Do_ps3.c
---------------------------------------------------------------
nlsp = 64       !   number of partial waves
---------------------------------------------------------------
001   0   2     1   p_2Se.c
002   0   4     1   no
003   0   2    -1   no
004   0   4    -1   p_4So.c
005   1   2     1   p_2Pe.c
006   1   4     1   p_4Pe.c
007   1   2    -1   p_2Po.c
008   1   4    -1   no
009   2   2     1   p_2De.c
010   2   4     1   no
011   2   2    -1   p_2Do.c
012   2   4    -1   no
013   3   2     1   no
..........................
063  15   2    -1   no
064  15   4    -1   no
---------------------------------------------------------------
```

Fig. 2. Example of input file **target** prepared by the user.

continuum orbitals are considered, if only the additional restrictions are not indicated by input parameters **max_ll**, **max_LT**, or **max_ST**. These parameters are mainly used in bound-state calculations, in order to get more compact expansions. Then, if there is a **pert_nnn.c** file for a perturber, this expansion is also added without any change. Finally, if some orthogonal conditions are imposed on the continuum orbitals, the program generates the additional $(N+1)$-electron configurations, which are intended to compensate for the orthogonality restraints imposed on the continuum orbitals. The resulting configuration expansion for each partial wave is recorded in the file **cfg.nnn**, and the relevant channel information is added to the file **target**.

The continuum configurations in the close-coupling expansions in the **cfg.nnn** files have the same expansion coefficients as the corresponding target states, whereas the expansion coefficients for the $(N+1)$-electron configurations are equal to unity. Further modification of the expansion coefficients is introduced in the case of intermediate coupling. The configurations in the c-files are written in consecutive LS-coupling of shells terms and have a definite total LS term. The intermediate coupling scheme is only applied to the coupling of the target with the continuum electron. So, in order to guarantee that the given configuration represents jK- or jj-coupling, we should apply the corresponding unitary transformation to the coupling between the target and the continuum electron. The transformation coefficients are given by the following formulas:

$$\begin{aligned} T_{LS,jK} &= \langle [(l_1l_2)L,(s_1s_2)S]J \,|\, [(l_1s_1)j_1,l_2]K,s_2,J\rangle \\ &= (-1)^{s_2+J-l_2-j_1}[L,S,j_1,K]^{1/2} \begin{Bmatrix} L & s_1 & K \\ s_2 & J & S \end{Bmatrix} \begin{Bmatrix} l_2 & l_1 & L \\ s_1 & K & j_1 \end{Bmatrix}, \end{aligned} \tag{5.1}$$

```
 e + C
------------------------------------------------------------------------
coupling = LS    !   LS coupling
nz = 6           !   nuclear charge
nelc = 6         !   number of electrons
------------------------------------------------------------------------
ntarg = 36       !   number of target states
------------------------------------------------------------------------
2p2_3P.c             targ_001.c    1   3    1     -37.77853957   258   10
2p2_1D.c             targ_002.c    2   1    1     -37.72880614   263    0
2p2_1S.c             targ_003.c    0   1    1     -37.67441594   132    0
2p3_5S.c             targ_004.c    0   5   -1     -37.62901126    30   10
2p3s_3Po.c           targ_005.c    1   3   -1     -37.50336422   318    0
2p3s_1Po.c           targ_006.c    1   1   -1     -37.49458261   261    0
2p3_3Do.c            targ_007.c    2   3   -1     -37.48150929   233    0
2p3p_1P.c            targ_008.c    1   1    1     -37.46513100   169    0
........................................................................
3Po_ps4.c            targ_035.c    1   3   -1     -36.72115047   318    0
3Do_ps3.c            targ_036.c    2   3   -1     -36.69897493   233    0
------------------------------------------------------------------------
nct = 8091
nwt = 20
  1s 2s1 2p1 3p1 3d1 3s1 9s1 9p1 9d1 9f1 2s2 2p2 3d2 3p2 3s2 4s2 9s2 9p2
 9d2 9f2
------------------------------------------------------------------------
nlsp =   64       !  number of partial waves
------------------------------------------------------------------------
001    0    2     1  p_2Se.c                pert_001.c     31    6
002    0    4     1  no                     no              0    0
003    0    2    -1  no                     no              0    0
004    0    4    -1  p_4So.c                pert_004.c     28    9
005    1    2     1  p_2Pe.c                pert_005.c     63    6
006    1    4     1  p_4Pe.c                pert_006.c     40    6
007    1    2    -1  p_2Po.c                pert_007.c     84    6
008    1    4    -1  no                     no              0    0
009    2    2     1  p_2De.c                pert_009.c     55    6
010    2    4     1  no                     no              0    0
011    2    2    -1  p_2Do.c                pert_011.c    141   10
012    2    4    -1  no                     no              0    0
013    3    2     1  no                     no              0    0
........................................................................
063   15    2    -1  no                     no              0    0
064   15    4    -1  no                     no              0    0
------------------------------------------------------------------------
```

Fig. 3. Example of the file **target** after a run of the BSR_PREP program.

$$T_{LS,jj} = [L,S,j_1,j_2]^{1/2} \begin{Bmatrix} l_1 & l_2 & L \\ s_1 & s_2 & S \\ j_1 & j_2 & J \end{Bmatrix}. \tag{5.2}$$

Here the indices 1 and 2 denote the target and continuum orbital, respectively; LS is the total term of the given configuration; J is the total angular momentum for the partial wave, and $[L]$ stands for $(2L+1)$. The expansion coefficients for the continuum configurations in the files **cfg.nnn** are multiplied by the corresponding transformation coefficients, depending on the intermediate coupling scheme under consideration. This procedure allows us to run all further calculations for generating and diagonalizing the Hamiltonian matrix in the same way for all coupling schemes.

The most complicated point concerns the orthogonality constraints imposed on the continuum orbitals. The present package allows the user to run calculations in different orthogonality modes, including the case when all orbitals are considered orthogonal as in the standard R-matrix codes. The biggest advantage, however, can be gained by releasing some orthogonality constraints. The orthogonality mode is defined by the input parameter **jort**. The default mode (**jort** $=$ **1**) is a partial orthogonality when the continuum orbitals are orthogonal to a restricted number of bound orbitals and means the following:

(a) All continuum orbitals should be orthogonal to the closed-shell orbitals that are common to all states. Such shells will also be referred to as core shells. Since these shells are closed, this does not lead to any additional $(N+1)$-electron bound terms in the close-coupling expansion in order to compensate these restrictions. The orthogonality of the scattering orbitals to closed-shell core functions is physically justified. It avoids unphysical low-lying R-matrix poles corresponding to the capture of the incident electron into these shells. Orthogonality to the closed core shell is automatically imposed in the present package, without any additional input parameters.

(b) Some shells can be considered as 'almost' closed if their average occupation number in the given target expansion is close to $(4l+2)$. For example, a configuration expansion for the ground state of neon, $1s^22s^22p^6$, may contain only a few correlated configurations with an excited 2s electron, and these configurations have relatively small expansion coefficients. In this case we also can impose the orthogonality of the *ks* continuum electron to the 2s orbital without having to introduce additional compensation configurations. Such cases are regulated by the input parameter **c_orb**. A shell is considered effectively closed, when the average occupation number for the orbital is greater than **c_orb** $\times (4l+2)$.

In the partial orthogonality mode, **jort** $=1$, all additional orthogonality conditions should be indicated as $\langle \boldsymbol{kl}|\boldsymbol{nl}\rangle = \mathbf{0}$. These conditions can be placed in the input file **bsr_par** and they will be effective for all partial waves, or they can be placed at the end of the **cfg.nnn** file and then will only be effective for a given partial wave. If we introduce some orthogonal constraint, we should generally also introduce the relevant $(N+1)$-electron bound states in order to compensate for this constraint in the full functional space. If we have a continuum configuration such as 'target, *kl*' and the continuum orbital *kl* is imposed to be orthogonal to the bound orbital *nl*, then the compensation configurations have the structure 'target, *nl*', with proper recoupling for the equivalent electrons. In doing this, however, we lost the information about the expansion coefficients for the individual target configurations. As practical calculations show, not all target configurations here are of the same importance. The compensation configurations based on the target configurations with small expansion coefficients may lead to an overestimate of the compensation effects. These, in turn, may lead to overestimating the cross sections or to additional pseudo-structure. To avoid such situations, we introduce the parameter **c_orth**, so that compensation configurations will only be generated for target states with expansion coefficients greater than **c_orth**.

Another important situation is related to the case when different channels can generate the same configurations. Consider again, for example, electron scattering from neon, with channels $2p^6kl$ and $2p^5nlkp$. If the orbital kl has no orthogonal constrains, then this channel can generate the $2p^6nl$ states. The same states will also be generated by second channel if the kp orbital is not orthogonal to the 2p orbital. Such situation may lead to 'overloaded' overlap matrix in the generalized eigenvalue problem (2.17). As a result, the overlap matrix will no longer be positive definite, and this prevents its diagonalization. To avoid such a situation, the user should impose the orthogonality constraint $\langle k\mathrm{p}|2\mathrm{p}\rangle = 0$ in the given example, without any compensation configurations, because these configuration can still can be generated in the first channel. As a general advice, we suggest to impose all orthogonality constraints which do not lead to compensation configurations. (The parameter **icomp** should be zero in this case.) It also simplifies the generation of the Hamiltonian matrix and, in addition, avoids the numerical instability in the diagonalization of the generalized eigenvalue problem (2.17).

Another important remark concerns the cases when orthogonality constraints are imposed to non-orthogonal but very similar orbitals. For example, we can use term-dependent radial functions for the same *nl* orbital, or we can introduce relaxation effects in the core-excitation. In the above example of e–Ne scattering, we can use two different radial functions for the 2p orbital in the $2p^5$ and $2p^6$ configurations to account for the relaxation effects. These functions are different but very similar. In this case we should apply the $\langle k\mathrm{p}|2\mathrm{p}\rangle = 0$ condition only to one of these orbitals.

5.2. Data files

bsr_par	File type: formatted sequential input. Written by user. Description: input parameters for given run. Format: see Section 3.2 and examples in Section 14.
target	File type: formatted sequential input. Written by user. Read and modified by programs BSR_PREP and BSR_CONF. Description: contains list of target states and scattering channels. Format: see Section 4.3 and Section 5.4 below.
targ_nnn.c	File type: formatted sequential input.

	Created by program BSR_PREP. Read by program BSR_CONF. Description: contains configuration expansion for target states **nnn**.
pert_nnn.c	File type: formatted sequential input. Created by program BSR_PREP. Read by program BSR_CONF. Description: contains configuration expansion for perturber **nnn**.
cfg_nnn.c	File type: formatted sequential output. Created by program BSR_CONF. Read by programs BSR_BREIT and BSR_MAT. Description: contains configuration expansion for partial wave **nnn**.
bsr_conf.log	File type: formatted sequential output. Written by program BSR_CONF. Read by user. Description: running information.

5.3. Input parameters

Input parameters can be provided in the command line or in the input file **bsr_par** (data from the command line overwrite data from the input file). Below we describe the data from **bsr_par** that are read by the program BSR_CONF. The default values for all data are indicated in the brackets.

jort [1]	orthogonality mode:
	jort < 0 – full orthogonality;
	jort $= 0$ – full non-orthogonality;
	jort > 0 – partial orthogonality, default.

icomp [0]	if $= 1$, the compensation configurations will be generated.
c_orth [0.75]	tolerance for generation of compensation configurations, which will be generated only for target states with expansion coefficients greater than **c_orth**.
c_orb [0.95]	tolerance for closed shells. The continuum orbitals are supposed to be orthogonal to closed shells, and also to almost closed shells, when the average occupation number for the orbital is greater than **c_orb** $\times (4l+2)$.
max_ll [-1]	restriction on the orbital angular momentum for the continuum orbitals (the default value of -1 means no restrictions).
max_LT [-1]	restriction on the total orbital angular momentum for the partial waves (the default value of -1 means no restrictions).
max_ST [-1]	restriction on the total spin for the partial waves (the default value of -1 means no restrictions).
$\langle \boldsymbol{kl} \vert \boldsymbol{nl} \rangle = \mathbf{0}$	additional orthogonality condition for the continuum orbital kl. If a set index indicated, this orthogonality condition will be applied to specified orbitals; otherwise, it will be applied to all orbitals with the given l. By default, all continuum orbitals are supposed to be non-orthogonal to the bound orbitals.

An additional input parameter **coupling** is read from file **target**. This parameter has three possible values **LS**, **JK** and **JJ**, and defines the coupling for continuum states in the case of relativistic calculations. In this case, the input target states should be provided for each J level, with the J-value being indicated in the first line of the corresponding c-file as $2J = value$.

5.4. Modification of file ***target***

BSR_CONF modifies the file **target** by adding information about the close-coupling expansions generated. An example of such information is given in Fig. 4. First, for each partial wave, the program provides the number of channels, **nch**, and the number of configurations, **nc**, in the corresponding **cfg.nnn** file. The number of configurations is represented by two values, corresponding to the continuum and the $(N+1)$-electron parts of the given close-coupling expansion. Then, for each channel in a separate line, we have the spectroscopic notation for the continuum orbital, its orbital momentum l, a pointer to the corresponding target state,

```
--------------------------------------------------
channels:
--------------------------------------------------
 1.    001    nch =   22    nc =    5437    31
kd4      2      2      1             263     0
ks4      0      3      2             395     0
kp4      1      5      3             713     0
kp5      1      6      4             974     0
...................................................
kpf      1     35     22            5406     0
---------------------------------------------------
 2.    002    nch =   11    nc =    3065     0
kp3      1      5      1             318     0
kd3      2      9      2             600     0
ks3      0     10      3             704     0
...................................................
kpa      1     35     11            3065     0
---------------------------------------------------
 3.    003    nch =   13      nc =  2655     0
kp3      1      1      1             258     0
kd3      2      7      2             491     0
kp4      1      8      3             660     0
...................................................
kd9      2     36     13            2655     0
---------------------------------------------------
...................................................
---------------------------------------------------
 64.   064    nch =   37     nc =   9689     0
ku1     15      1      1             258     0
kt1     14      5      2             576     0
kv1     16      5      3             894     0
kt2     14      7      4            1127     0
...................................................
kve     16     36     37            9689     0
---------------------------------------------------
max_ch  =          66
max_nc  =       15921
max_wf  =          90
---------------------------------------------------
```

Fig. 4. Channel information added to **target** by the program BSR_CONF.

index of the channel, a pointer to the last configuration for the given channel in the file **cfg.nnn**, and the value of K, if we use the jK intermediate coupling scheme. The spectroscopic notation for the continuum orbital is given only for the user's information. As additional information, the end of **target** provides the maximum number of channels, the maximum number of configurations, and the maximum number of different one-electron orbitals used in the close-coupling expansion.

6. Program BSR_BREIT

6.1. Outline of the BSR_BREIT calculations

The BSR_BREIT program performs the angular integrations necessary to express the matrix elements of the Breit–Pauli Hamiltonian as a linear combination of radial integrals. Any amount of non-orthogonality between the orbitals may be present, leading to overlap factors in the matrix elements. The program can effectively reuse data obtained previously by creating a databank for angular coefficients. BSR_BREIT is a new optimized version of the general program BREIT_NO [24]. It can be used in many other aspects of atomic-structure calculations. The calculations of angular integrals follow the method based upon the representation of configuration wave functions through Slater determinants. The details of the present approach can be found in the long write-up of the program BREIT_NO. Below we only provide a short description of the present version, outlining the most significant modifications made in comparison to the original BREIT_NO.

The matrix element in the most general case of fully non-orthogonal orbitals has the form

$$\big\langle \Phi\big(\{nl\}\gamma LSJ\big)\big|\mathrm{O}\big|\Phi'\big(\{n'l'\}\gamma' L'S'J'\big)\big\rangle = \sum a_k R^k(i,j;i',j') \times D\big(\{nl\};\{n'l'\}\big), \tag{6.1}$$

where a_k are the numerical coefficients, which depend only on the angular symmetry of the configuration state functions (CSFs) involved, R^k is the corresponding radial integral dependent on the operator **O** under consideration, and $D(\{nl\},\{n'l'\})$ is the overlap factor, which depends only on the sets of radial orbitals used in the construction of the wavefunctions Φ and Φ', respectively. Hereafter, the notation *angular symmetry* (AS) is used to denote the full set of angular momenta $(\{l\}\gamma LS(L'S'))$, which includes the momenta of spin–orbitals $\{l\}$, the terms of subshells $\alpha_i L_i S_i$, and the intermediate terms $L'_i S'_i$ as well as the total momenta LS. The symbol γ defines the coupling of momenta of all occupied subshells according to the given scheme

$$\Big(\big(\big((\text{core}, L_1)L'_1, L_2\big)L'_2, L_3\big)L'_3\ldots\Big)L, \quad \Big(\big(\big((\text{core}, S_1)S'_1, S_2\big)S'_2, S_3\big)S'_3\ldots\Big)S, \quad \boldsymbol{L}+\boldsymbol{S}=\boldsymbol{J}. \tag{6.2}$$

It is convenient to introduce the notation *generalized shell term* for $(\alpha L\,S)\,L'S'$, which also includes the corresponding intermediate momenta $L'S'$. It should be emphasized that the coefficients a_k do not depend directly on the set of the principal quantum numbers $\{n\}$ and $\{n'\}$. The overlap factor $D(\{nl\},\{n'l'\})$ is generally a product of determinants corresponding to the matrices of one-electron overlap integrals $\langle nl|n'l\rangle$; it depends only on the orthogonality conditions imposed on the one-electron radial functions. For example, in the special case of orthogonal radial orbitals, the overlap factors are trivially reduced to 1 or 0, and we obtain the weighted sum of radial integrals R^k, which is ordinarily used in atomic structure calculations. For partial orthogonality conditions, some D-factors disappear while others are simplified. Thus, once we obtain the matrix element in the case of fully non-orthogonal orbitals, we can also obtain the matrix elements for all other configurations with the same AS and for all other orthogonal conditions, only by analyzing the overlap factors $D(\{nl\},\{n'l'\})$ and replacing the set $\{n,n'\}$ by the new one. Such a scheme is realized in the present code. In large-scale calculations, it can lead to a considerable reduction of the execution time, because in practical calculations the set of configurations contains many configurations with the same angular symmetry.

On the other hand, rather than record the angular coefficients a_k for a given set of CSFs, we can save only the expansions for the matrix between ASs with full non-orthogonality conditions. This data may be considered as a *databank* for angular coefficients a_k. They allow us to obtain the expansions of matrix elements for any set of CSFs constructed with the ASs included in the databank. In case of new symmetries, we have to perform only the additional calculations and thus the databank can be easily extended.

The next idea concerns the structure of the angular coefficients a_k themselves. Because the calculations of angular integrals follow the method based upon the representation of configuration wavefunctions through Slater determinants, the coefficients a_k can be separated into a factor that depends only on the Slater determinants involved and a factor dependent on the coupling scheme (6.2). Since many atomic states contain the same Slater determinants, it seems to be much more efficient to first calculate the matrix elements between separate Slater determinants before multiplying them with the coupling-scheme factor depending on the AS under consideration. The Slater determinants involved are defined only by the sets $\{l^q\}$ where q is the shell occupation number. Hereafter, we will refer to the set $\{l^q\}$ as a *configuration symmetry* (CS). Hence, in order to avoid the recalculation of matrix elements between the same Slater determinants, it is most effective to first define the list of all possible Slater determinants and then to obtain the angular coefficients for the matrix elements between these Slater determinants.

The above ideas have been implemented in the present code. It requires additional sorting of the input configuration list according to their ASs as well as to CSs. The principal features of the present version are given below.

6.2. Structure and data flow

The block diagram of the program BSR_BREIT, along with the data flow, is shown in Fig. 5. The only input data file is a c-file with a list of the configurations under consideration. The default name for this file is **cfg.inp**, or **cfg.nnn**, if a set of c-files for different partial waves **nnn** should be handled. The range of required partial waves is given by the input parameters **klsp1** and **klsp2** (see Section 6.4). BRS_BREIT requires only a limited number of input parameters, most of which can be defined in the command line in the format ‘*name* = *value*’. There is only one optional position parameter **case.c** which defines the name of specific input c-file. Another input parameter is **oper**, which defines a set of operators from the Breit–Pauli Hamiltonian to be considered. All input parameters are read from the command line by the routine **read_arg**.

Then the program executes fully independent calculations for each partial wave. First, the **R_conf** routine reads the input configurations in spectroscopic notation, decodes them to the internal ‘integer’ representation and tests the CSFs as to the correctness of angular coupling, the number of electrons, parity and AFTER conditions. **R_conf** also allocates (or reallocates) all the main arrays with dimensions specific for a given partial wave. Then the CSFs are sorted according to their angular and configuration symmetries. At this stage the program also checks if it is a continued calculation, i.e. if there exists a relevant databank from previous calculations. The program analyses information contained in the databank and defines the subset of configurations and operators that need additional consideration, or it decides that information in the databank is sufficient for the given input configuration list.

In the next block, **pre_detexp**, the determinant expansions for all configurations are generated and recorded in a scratch file. This block is the main modification made in comparison to the previous version, BREIT_NO. It allows one to avoid a large amount of recalculations for repeating determinant coefficients. The determinant expansion for a given angular symmetry is generated in the routine **det_expn**, on the basis of angular coupling coefficients, calculated in **detc_sh**, and coefficients for determinant expansions of individual shells, stored in the blockdata **blk_det**.

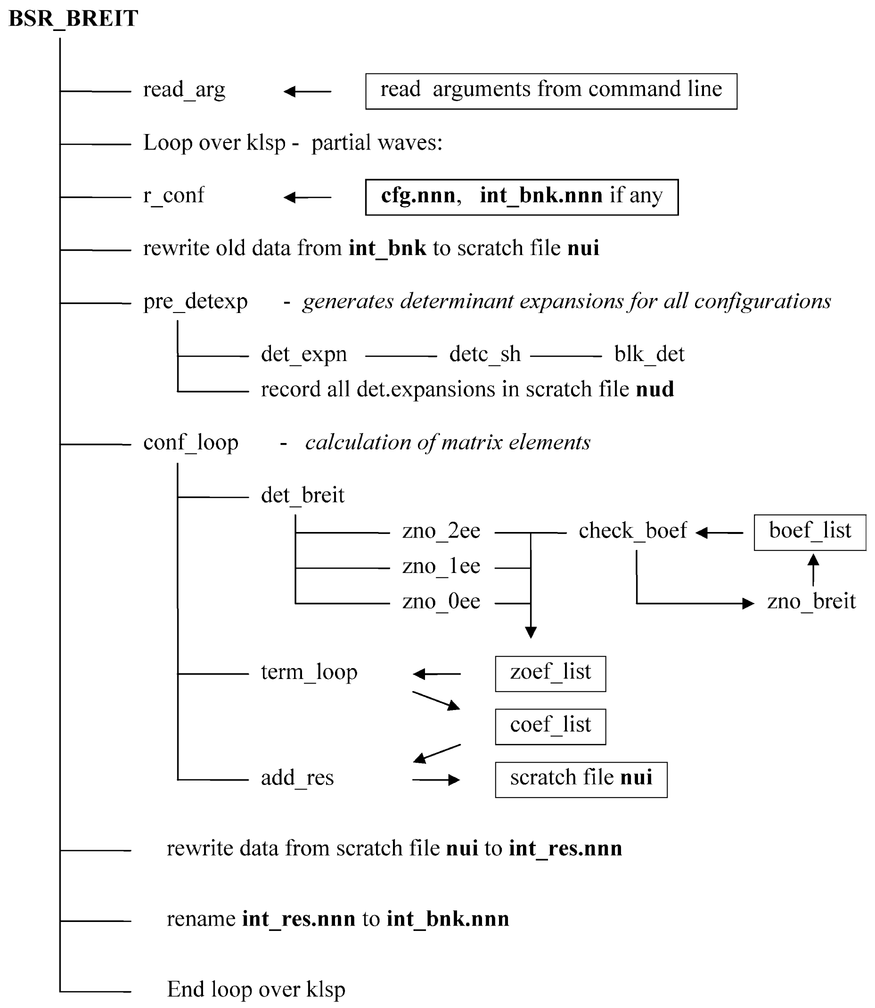
BSR_BREIT
read_arg
read arguments from command line
Loop over klsp - partial waves:
r_conf
cfg.nnn, int_bnk.nnn if any
rewrite old data from int_bnk to scratch file nui
pre_detexp - generates determinant expansions for all configurations
det_expn
detc_sh
blk_det
record all det.expansions in scratch file nud
conf_loop - calculation of matrix elements
det_breit
zno_2ee
zno_1ee
zno_0ee
check_boef
boef_list
zno_breit
term_loop
zoef_list
coef_list
add_res
scratch file nui
rewrite data from scratch file nui to int_res.nnn
rename int_res.nnn to int_bnk.nnn
End loop over klsp

Fig. 5. Block diagram for the program BSR_BREIT and data flow (see text).

The calculations of a full set of angular coefficients and overlap factors for input angular symmetries are running in the block **conf_loop**. First, routine **det_breit** computes the angular coefficients for matrix elements between all Slater determinants involved. This information is accumulated in the module **zoef_list**. The cases of overlaps, one-electron and two-electron operators are treated separately by the routines **zno_0ee**, **zno_1ee**, and **zno_2ee**, respectively. Routine **zno_breit** contains expressions for all Breit–Pauli two-electron matrix elements in the *nlms*-representation. All these expressions are presented in the BREIT_NO write-up [24]. The corresponding angular coefficients for the integrals involved are accumulated in the module **boef_list**, in order to avoid their recalculation. It is controlled by the routine **check_boef**.

Then, in routine **term_loop**, the angular coefficients between two given angular symmetries are calculated based upon the matrix elements between the determinants involved, stored in module **zoef_list**, and the determinant expansion coefficients, which are read from the scratch file **nud** (see block **pre_detesp)**. These data are accumulated in the module **coef_list**. When all determinants for the given angular symmetries are exhausted, the final angular coefficients are recorded by the routine **add_res** in the scratch file **nui**, and module **coef_list** is initialized for accumulation of data from other matrix elements. The above calculations are repeated in the block **conf_loop** for all combinations of input angular symmetries.

Finally, the new data and data from previous calculations are stored in file **int_res.nnn**. This file is then renamed to the initial name **int_bnk.nnn**. Such a procedure prevents loosing the data in the initial databank if an error occurs when running BSR_BREIT. The format of the output databank is given in Section 6.5. The output databank has the default name **int_bnk** for default input *c*-file **cfg.inp**. For the specific input *c*-file **case.c**, the output databank has the name **case**.**bnk**.

All calculations in BSR_BREIT are carried out in full non-orthogonal mode, without specifications of principal quantum numbers for the orbitals involved. Instead, the positions of the corresponding shells in the configurations are used to specify the one-electron orbitals involved. Consequently, the data obtained can be used further for all atomic states with the given angular symmetries. In the present version we also abandoned the creation of an additional list of angular coefficients for the given set of one-electron orbitals (the file **int.lst** in BREIT_NO). Instead, the matrix elements for specific orbitals are calculated dynamically at the stage when the Hamiltonian matrix is generated in the program BSR_MAT. This approach has two benefits. First, we do not loose the time for sorting the data in the **int.lst** file. Second, we save disk space, because the **int_bnk** files are usually much smaller.

6.3. Data files

cfg.nnn	File type: formatted sequential input. Created by program BSR_CONF. Read by programs BSR_BREIT and BSR_MAT. Description: contains configuration expansion for partial wave **nnn**.
int_bnk.nnn	File type: unformatted sequential output. Written by program BSR_CONF. Read by BRS_MAT. Description: databank for angular coefficients.
bsr_breit.log	File type: formatted sequential output. Written by program BSR_BREIT. Read by user. Description: running information.

6.4. Input parameters

Input parameters can be provided only in the command line in the format '*name* = *value*'. All data have the default values indicated in the brackets.

klsp1 [0]	first partial wave under consideration; default value **klsp1** = 0 means that input configuration file is **cfg.inp** and output data will be placed in file **int_bnk**.
klsp2 [klsp1]	last partial wave under consideration.
oper [1110000]	character*7, where each position can be 0 or 1, and indicate the operator under consideration: oper(1) – overlaps oper(2) – kinetic energy oper(3) – two-electron electrostatic oper(4) – spin–orbit oper(5) – spin-other-orbit oper(6) – spin–spin oper(7) – orbit–orbit Default value **`oper`** = 1110000 stands for non-relativistic calculations.
mk [9]	maximum multipole index.
mrecl [100000]	maximum record length (in bytes) in unformatted files. This parameter was introduced to insure the portability of the unformatted files on different computers and platforms.

6.5. Structure of the angular-coefficients databank

Summary of the output records in the **int_bnk** file.

1. `nsymt,nsymc`

   `nsymt` – number of different angular symmetries
   `nsymc` – number of different configuration symmetries

2. `ASYM(1:nsymc)`

   List of configuration symmetries (character*26). Each configuration symmetry `ASYM(.)` contains `(l(i),q(i),i=1,no),LS` in format `(8(a1,i2),2a1)`, where `no` – number of shells; `l(i)` – electron orbital momentum in `i`th shell; `q(i)` – number of electrons in `i`th shell; `LS` – total term in spectroscopic notation.

3. `BSYM(1:nsymt)`

List of angular symmetries (character*40). Each angular symmetry `BSYM(.)` contains a list of the generalized terms of shells `(A,L,S, LI,SI, 1:no)` in format `(8(i1,a1,2i1,a1))`. Each generalized term includes: `A` – seniority number; `L` – orbital momentum of the shell; `S` – spin; `LI` – intermediate orbital momentum; `SI` – intermediate spin.

4. `IT_CONF(1:nsymt)`

   Indicates the corresponding configuration symmetry for a given angular symmetry, i.e. it defines the relation between the `ASYM` and `BSYM` arrays.

5. `nmatr`
6. `IT_OPER(1:7,1:nmatr)`

   `nmatr = nsymt` $\times$ `(nsymt+1)/2` – dimension of the `IT_OPER` array.
   `IT_OPER(.,ij)` indicates the operators that have already been considered by the program for angular symmetries `i,j` such that `ij = i` $\times$ `(i+1)/2+j, i > j`.
   `IT_OPER(1:7,.)` indicates the operator under consideration, in the same way as the input parameter **oper** (see Section 6.4).

7. `ndet,kdet`
8. `KPD(1:ndet)`
9. `IPD(1:ndet)`
10. `NPD(1:kdet)`

   `ndet` – total number of overlap determinants.
   `kdet` – size of the NPD array, containing the description of all overlap determinants.
   `KPD(i)` – dimension of the `i`th overlap determinant, `kd`.
   `IPD(i)` – pointer on the `i`th overlap determinant in array `NPD`, `ip`.
   `NPD(ip+1:ip+kd)` – contains the description of the `i`th determinant as `NPD(.)` $= i_1 * b_d + i_2$, where $b_d = 2^{15}$ – packing basis for overlap determinants; i_1, i_2 – pointers to shells in the involved configurations.

11. `ndef,kdef`
12. `KPF(1:ndef)`
13. `IPF(1:ndef)`
14. `NPF(1:kdef)`

   `ndef` – number of different overlap factors.
   `kdef` – size of the NPF array, containing the description of all overlap factors.
   `KPF(i)` – number of determinants in the `i`th overlap factor, `kd`.
   `IPF(i)` – pointer on the `i`th overlap factor in array `NPF`, `ip`.
   `NPF(ip+1:ip+kd)` – contains the description of the `i`th overlap factor as `NPF(.)` $= ipd * b_f + nd$, where $b_f = 16$ – packing basis for overlap factors; `ipd` – pointer to the overlap determinant; `nd` – its power.

15. `C,ij,int,idf` repeat up to the end of file

   `C` – angular coefficient for matrix element between angular symmetries `i` and `j`, recorded as $ij = i \times b_c + j$, where $b_c = 2^{15}$ – packing basis for configurations.
   `int` – pointer on the relevant radial integral in the packing form $int = m \times b^8 + k \times b^4 + i_1 \times b^3 + i_2 \times b^2 + i_3 \times b + i_4$, where `m` – type of integral; `k` – multipole index; i_1, i_2, i_3, i_4 – pointer on the involved orbitals; $b = 10$ – packing basis.
   `idf` – pointer on the corresponding overlap factor.

NOTE: all long arrays are divided into records with maximum length **mrecl**, in order to avoid a stack overflow on different platforms.

7. Program BSR_MAT

7.1. Outline of the BSR_MAT calculation

This program generates the interaction matrices (2.17) in the B-spline representation. It can be done either in LS-coupling or in some intermediate coupling scheme with direct inclusion of selected Breit–Pauli terms. The eigenproblem (2.17) in the B-spline basis leads to a generalized eigenvalue problem of the form

$$\boldsymbol{Hc} = E\boldsymbol{Sc}. \tag{7.1}$$

The solution vector, $\boldsymbol{c}$, can be written as

$$\boldsymbol{c} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{N_c}, \boldsymbol{b}]^{\mathrm{T}}, \tag{7.2}$$

where each $\boldsymbol{a}_i$ is a column vector of B-spline coefficients for the given channel functions $\bar{\Phi}_i^{\Gamma}$

$$\boldsymbol{a}_i = [a_{1i}, a_{2i}, \ldots, a_{n_s i}]^{\mathrm{T}} \tag{7.3}$$

and $\boldsymbol{b}$ is the column vector of correlation functions coefficients

$$\boldsymbol{b}_j = [b_1, b_2, \ldots, b_{N_p}]^{\mathrm{T}}. \tag{7.4}$$

Here N_c is the number of channels while N_p is the number of correlation functions. Schematically, the interaction matrix $\boldsymbol{H}$ has the form

$$\begin{pmatrix} H(11) & H(12) & \ldots & H(1N_c) & h(1p) \\ H(21) & H(22) & \ldots & H(2N_c) & h(2p) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ H(N_c1) & H(N_c2) & \ldots & H(N_cN_c) & h(N_cp) \\ h(1p)^{\mathrm{T}} & h(2p)^{\mathrm{T}} & \ldots & h(N_cp)^{\mathrm{T}} & h(pp) \end{pmatrix}, \tag{7.5}$$

where $h(pp)$ is an $N_p \times N_p$ matrix that comes from the bound–bound interaction, the $h(ip)$ are $n_s \times N_p$ matrices representing the interaction between the ith channel and the bound states, and $H(ij)$ are $n_s \times n_s$ matrices for channel–channel interaction. Furthermore, $\boldsymbol{S}$ in Eq. (7.1) is the overlap matrix. In the usual case of orthogonal conditions imposed on the scattering orbitals, it reduces to a matrix consisting of diagonal blocks of the banded overlap matrix $\boldsymbol{B}$ between individual B-splines B_i

$$B_{ij} = \langle B_i | B_j \rangle, \tag{7.6}$$

one for each channel. In the more general case of non-orthogonal orbitals, however, it has a more complicated structure (see Table 1).

Each individual matrix element in (7.5) is expressed by the program BSR_BREIT in the form

$$\sum c_i L(a,b) \times D\big(\{nl\}; \{n'l'\}\big) + \sum c_i R^{\lambda}(a,b;c,d) \times D\big(\{nl\}; \{n'l'\}\big), \tag{7.7}$$

where the c_i are numeric coefficients which depend only on the angular symmetry of the CSFs involved, L stands for one-electron integrals, R^{λ} are Slater integrals, and $D(\{nl\}, \{n'l'\})$ is the overlap factor, which depends only on the orthogonality of the radial orbitals used in the construction of the CSFs. In general, the overlap factor is the multiplier of determinants of matrices consisting of one-electron overlaps

$$\begin{vmatrix} \langle n_1l|n_1'l\rangle & \langle n_1l|n_2'l\rangle & \ldots & \langle n_1l|n_q'l\rangle \\ \langle n_2l|n_1'l\rangle & \langle n_2l|n_2'l\rangle & \ldots & \langle n_2l|n_q'l\rangle \\ \ldots & \ldots & \ldots & \ldots \\ \langle n_ql|n_1'l\rangle & \langle n_ql|n_2'l\rangle & \ldots & \langle n_ql|n_q'l\rangle \end{vmatrix}. \tag{7.8}$$

The scattering orbitals can appear in the radial integrals L and R^{λ}, and also in the determinant factors. To derive the final expression for the matrix elements in the B-spline basis, let us first simplify the overlap factors (7.8) by expanding them over the row (or column) that contains one-electron overlaps with the scattering orbitals. At most, there can be one such row or/and column. The residual overlap determinants depend only on the known bound orbitals and they can be calculated in a standard manner. It is convenient to redefine the angular coefficients c_i, multiplying them by these residual overlap factors. As a result, the initial overlap factor $D(\{nl\}, \{n'l'\})$ in expression (7.7) is reduced to one- or two-electron overlaps of the form

$$\langle kl|nl\rangle, \qquad \langle kl|k'l\rangle \quad \text{or} \quad \langle kl|nl\rangle\langle n'l'|k'l'\rangle, \tag{7.9}$$

where $|kl\rangle$ stands for the radial function of the continuum orbital in the inner region

$$u_{kl}(r) = \sum_i a_i B_i(r). \tag{7.10}$$

Each term in (7.7) gives rise to a matrix in the spline basis. To elucidate the structure of these matrices, let us introduce, in addition to the overlap B-matrix defined in Eq. (7.6), $n_s \times n_s$ matrices with elements

$$\begin{aligned} &L(..)_{ij} = \langle B_i | \frac{\mathrm{d}^2}{\mathrm{d}r^2} - \frac{l(l+1)}{r^2} + \frac{2Z}{r} | B_j \rangle, \\ &R^{\lambda}(.ab.)_{ij'} = \sum_{i'j} a_{i'} b_{j'} R^{\lambda}(ij;i'j'), \qquad R^{\lambda}(.a.b)_{ij} = \sum_{jj'} a_{i'} b_{j'} R^{\lambda}(ij;i'j'), \end{aligned} \tag{7.11}$$

where

$$R^{\lambda}(ij,i'j') = \big\langle B_i(r_1)B_j(r_2)\big|U_{12}^{\lambda}\big|B_{i'}(r_1)B_{j'}(r_2)\big\rangle. \tag{7.12}$$

Here U_{12}^{λ} is the radial part of the λth term in the multipole expansion of $1/r_{12}$, and a_i, b_j are the expansion coefficients in the spline basis for the bound orbitals $P_a(r), P_b(r)$ as in (7.10). The four-dimensional array, $R^{\lambda}(ij;i'j')$, effectively represents the Slater integral in the B-spline basis. The integrals are zero if either $|i-i'| \geqslant k_s$ or $|j-j'| \geqslant k_s$. Methods for computing these quantities, which depend only on the basis, are given in Section 12. In order to describe the channel-bound interaction, let us introduce the following vectors with elements

$$B(.a)_i = \sum_{j=1}^{n_s} a_j B_{ij}, \tag{7.13}$$

$$L(.a)_i = \sum_{j=1}^{n_s} a_j L_{ij}, \tag{7.14}$$

$$R^{\lambda}(.abc)_i = \sum_{ji'j'=1}^{n_s} a_j b_{i'} c_{j'} R^{\lambda}(ij;i'j'). \tag{7.15}$$

With this notation, Table 1 lists the contributions of the different terms from the energy expression (7.7), which is usually produced by the angular integration codes, into the interaction matrix (7.5), along with an indication of its structure. Table 1 also indicates the contribution to the total overlap matrix arising from the matrix element on the right-hand side of Eq. (7.1). Note that the target

Table 1
Contribution of different terms (without indication of angular coefficients) in the interaction matrix. The symbols a, b, c, d stand for the bound orbitals, while k_i indicates the continuum orbital in channel i

Term	Contribution	Remarks
Channel–channel interaction		
$R^{\lambda}(a,k_i;b,k_j)$	$R^{\lambda}(a.b.)$	Banded matrix,
$R^{\lambda}(k_i,a;k_j,b)$	$R^{\lambda}(.a.b)$	direct interaction
$R^{\lambda}(k_i,a;b,k_j)$	$R^{\lambda}(.ab.)$	Full matrix,
$R^{\lambda}(a,k_i,;k_j,b)$	$R^{\lambda}(a..b)$	exchange interaction
$R^{\lambda}(k_i,a;b,c)\ \langle d\|k_j\rangle$	$R^{\lambda}(.abc).B(.d)$	Full matrix,
$R^{\lambda}(a,k_i;b,c)\ \langle d\|k_j\rangle$	$R^{\lambda}(a.bc).B(.d)$	contribution due to non-orthogonality
$R^{\lambda}(a,b;k_i,c)\ \langle d\|k_j\rangle$	$R^{\lambda}(ab.c).B(.d)$	
$R^{\lambda}(a,b;c,k_i)\ \langle d\|k_j\rangle$	$R^{\lambda}(abc.).B(.d)$	
$\langle k_i\|a\rangle\langle b\|k_j\rangle R^{\lambda}$ (or L)	$B(.a).B(.b) * R^{\lambda}$ (or L)	Full matrix, contribution due to non-orthogonality
$\langle k_i\|k_j\rangle R^{\lambda}$ (or L)	$B(..) * E_i$	Banded matrix, only for $i=j$
$L(k_i,k_j)$	$L(..)$	Banded matrix, only for $i=j$
$L(k_i,a)\langle b\|k_j\rangle$	$L(.a).B(.b)$	Full matrix, contribution due to non-orthogonality
$\langle k_i\|k_j\rangle$	$B(..)$	Banded matrix, only for $i=j$, contribution to overlap matrix
$\langle k_i\|a\rangle\langle b\|k_j\rangle$	$B(.a).B(.b)$	Full matrix, contribution to overlap matrix due to non-orthogonality
Channel–bound interaction		
$R^{\lambda}(k_i,a;b,c)$	$R^{\lambda}(.abc)$	Vector
$R^{\lambda}(a,k_i;b,c)$	$R^{\lambda}(a.bc)$	
$R^{\lambda}(a,b;k_i,c)$	$R^{\lambda}(ab.c)$	
$R^{\lambda}(a,b;c,k_i)$	$R^{\lambda}(abc.)$	
$L(k_i,a)$	$L(.a)$	Vector
$\langle k_i\|a\rangle R^{\lambda}$ (or L)	$B(.a) * R^{\lambda}$ (or L)	Vector, contribution due to non-orthogonality
$\langle k_i\|a\rangle$	$B(.a)$	Vector, contribution to overlap matrix
Bound–bound interaction		
$R^{\lambda}(a,b;c,d)$		Scalar
$L(a,b)$		Scalar
c		Scalar, contribution to overlap matrix

wave functions are assumed to form an orthogonal set and diagonalize the target Hamiltonian. Hence, of all the matrix elements containing $\langle kl|k'l\rangle$, only the diagonal ones, with $k=k'$, contribute to the interaction matrix. The same concerns the matrix elements with $L(kl,k'l)$.

7.2. Structure and data flow

The block diagram of the program BSR_MAT, along with the data flow, is shown in Fig. 6. First, the program reads the data common to all partial waves. It includes the parameters for the calculation (routine **read_arg**), the B-spline parameters (**define_spline**), and the description of the target states **(read_target)**. Then the program executes fully independent calculations for each partial wave. First, the program reads the configuration list from the relevant c-file **cfg.nnn** and allocates all main arrays with dimensions specific for a given partial wave. The most critical dimension here is the size of the interaction matrix, which usually occupies most of the memory. The size of the interaction matrix is mainly defined by the product of the number of scattering channels and the number of B-splines, because usually the $(N+1)$-electron bound channels in the close-coupling expansion are either absent or their number is relatively small in the present approach. In order to speed up the calculation, we place the entire interaction matrix in the RAM memory. On the other side, this limits the number of scattering channels which we can include in the physical model by the size of the computer memory. (Note that virtual memory will not help much here.)

The following calculations of the matrix elements are performed for each type of integrals sequentially. This part is controlled by the routine **state_res**, whose flow diagram is placed at the bottom of Fig. 6. Each time this routine reads the relevant **int_bnk** file from the beginning, chooses the integrals under consideration and provides their contribution to the total interaction matrix.

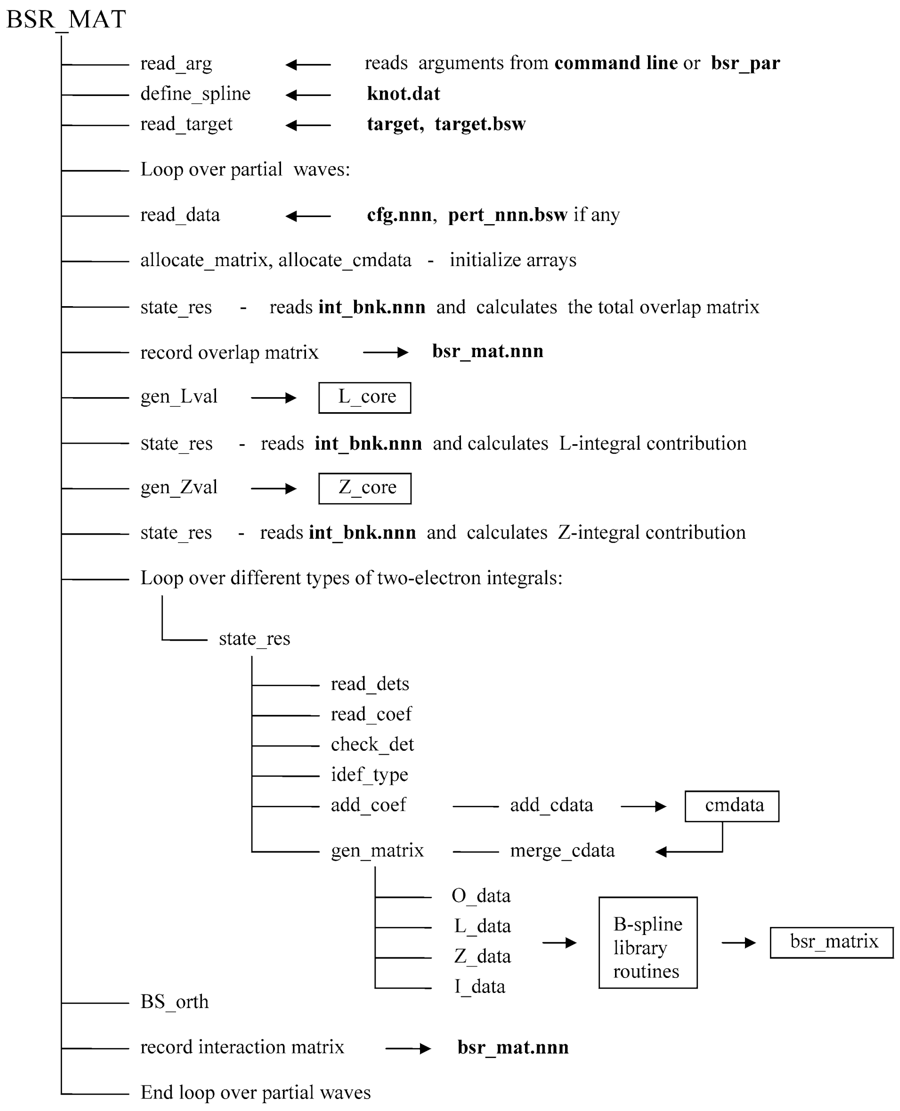
BSR_MAT
read_arg
reads arguments from command line or bsr_par
define_spline
knot.dat
read_target
target, target.bsw
Loop over partial waves:
read_data
cfg.nnn, pert_nnn.bsw if any
allocate_matrix, allocate_cmdata - initialize arrays
state_res - reads int_bnk.nnn and calculates the total overlap matrix
record overlap matrix
bsr_mat.nnn
gen_Lval
L_core
state_res - reads int_bnk.nnn and calculates L-integral contribution
gen_Zval
Z_core
state_res - reads int_bnk.nnn and calculates Z-integral contribution
Loop over different types of two-electron integrals:
state_res
read_dets
read_coef
check_det
idef_type
add_coef
add_cdata
cmdata
gen_matrix
merge_cdata
O_data
L_data
Z_data
I_data
B-spline library routines
bsr_matrix
BS_orth
record interaction matrix
bsr_mat.nnn
End loop over partial waves

Fig. 6. Block diagram for the program BSR_MAT and data flow (see text).

Such a procedure reduces the number of integrals which should be considered at one time, and this increases the effectiveness of their processing. One-electron and two-electron integrals are considered in different ways. For one-electron integrals, such as kinetic-energy L-integrals (7.11) or spin–orbit Z-integrals (2.47), the program first calculates all possible integrals and their B-spline representation for all one-electron orbitals in the configuration list, employing the routines **gen_Lval** and **gen_Zval**. The relevant data are stored in the modules **L_core** and **Z_core**. Their storing requires only two-dimensional arrays and can be handled even for a large number of one-electron orbitals. For the two-electron integrals, their storing requires four-dimensional arrays. This cannot be realized in the case of a large number of different one-electron orbitals, which often occur in the present approach. Hence, the two-electron integrals are calculated dynamically when they are required. In contrast to the standard R-matrix approach, the present implementation uses a larger number of non-orthogonal orbitals for a more accurate representation of the target states. Typical calculations may involve up to several hundred of different one-electron radial functions. The present code has no practical limit on the maximum number of different one-electron orbitals that may be used to describe the target and the scattering states.

The list of specific integrals needed for the given configurations is generated dynamically from the information in the databank **int_bnk**. The integrals are then sorted according to their structure (see Table 1) and stored in the module **cmdata** in different ordered blocks, using routines **check_det**, **idef_type, add_coef**, and **add_cdata**. The number and size of the individual blocks is defined by the input parameters and could be adjusted to the speed and memory of a given computer. When all blocks are full, they are merged, if necessary, with the routine **merge_cdata** and then are processed by the routine **gen_matrix** to get the B-spline representation for specific integrals (using routines **O_data**, **L_data**, **Z_data** and **I_data**; the latter deals with two-electron integrals). All these routines extensively use subroutines from the BSPLINE library. The corresponding contributions are then added to the interaction matrix, located in module **bsr_matrix**.

The next step is the correction of the interaction matrix according to the orthogonal conditions imposed on the scattering orbitals, using the routine **BS_orth**. This is done according to the procedure suggested by Bentley [55]. Suppose a continuum orbital, $u_{kl}(r)$, in some ith channel is orthogonal to the bound orbital $P_{nl}(r)$. This can be accounted for by modifying the Hamiltonian according to

$$\boldsymbol{H} \to (1 - \boldsymbol{B}\boldsymbol{c}\boldsymbol{c}^{\mathrm{T}})\boldsymbol{H}(1 - \boldsymbol{c}\boldsymbol{c}^{\mathrm{T}}\boldsymbol{B}), \tag{7.16}$$

where $\boldsymbol{c}$ is the full solution vector (7.2) with all zero elements, except that the channel i is replaced by the B-spline expansion $\{p_i\}$ of the orbital $P_{nl}(r)$. This leads to transformation of only those blocks $H(i,j)$ in (7.5) which have one index equal to i. Recall that the orthogonal conditions to the common core orbitals are implied automatically in the present implementation. All other conditions should be specified through input data in the **bsr_par** or **cfg.nnn** files.

At the last step, the interaction matrix, along with the long-range potential coefficients, is recorded in the relevant **bsr_mat.nnn** file. The above calculations are repeated for each partial wave under consideration.

7.3. Data files

bsr_par	File type: formatted sequential input. Written by user. Read by routine **read_arg**. Description: input parameters for given run.
target	File type: formatted sequential input. Written by user and modified by BSR_PREP and BSR_CONF programs. Read by routine **read_target**. Description: contains description of target states and scattering channels.
knot.dat	File type: formatted sequential input. Written by user. Read by routine **define_grid** from BSPLINE library. Description: input parameters that define the B-spline grid.
cfg.nnn	File type: formatted sequential input. Created by program BSR_CONF. Read by routine **read_conf**. Description: list of configurations for given partial wave **nnn**.

bnk_int.nnn	File type: unformatted sequential input. Created by program BSR_BREIT. Read by routine **state_res**. Description: databank for angular coefficients.
target.bsw	File type: unformatted sequential input. Created by program BSR_PREP. Read by routine **read_bsw**. Description: target's one-electron orbitals in B-spline basis.
pert_nnn.bsw	File type: unformatted sequential input. Created by program BSR_PREP. Read by routine **read_bsw**. Description: perturber's one-electron orbitals in B-spline basis, optional.
mat_log.nnn	File type: formatted sequential output. Written by program BSR_MAT. Read by user. Description: running information.
bsr_mat.nnn	File type: unformatted sequential output. Created by program BSR_MAT. Read by program BSR_HD. Description: Hamiltonian matrix and asymptotic coefficients.
int_mat.nnn	File type: formatted sequential output. Created by program BSR_MAT. Read by user. Description: list of one- and two-electron integrals. This output is only for debug purposes and controlled by the parameter **nud** in the **bsr_par** input file.

7.4. Input data

Input data can be provided in the command line or in the input file **bsr_par** (the data from the command line overwrite the data from input file). Below we describe those data from **bsr_par** which are read by program BSR_MAT. 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 **param_mat**.

klsp1 [1]	first partial wave under consideration.
klsp2 [klsp1]	last partial wave under consideration.
mk [7]	maximum multipole index.
mb [5000]	the size of one block in module **cmdata** used for the accumulation of a given type of integrals.
nb [500]	number of blocks in module **cmdata**.
mrel [0]	indicates the terms of the Breit–Pauli Hamiltonian which should be included `mrel=1` – include only one-electron scalar relativistic integrals. `mrel=2` – include additionally one-electron spin–orbit interaction. `mrel=3` – plus two-electron spin-other-orbit interaction. `mrel=4` – plus spin–spin interaction. `mrel=5` – plus orbit–orbit interaction.
mso [0]	controls the inclusion of one-electron spin–orbit interaction. If set to 0, then a decision will be made according to the **mrel** parameter; if set to $-1(1)$, then the spin–orbit interaction will be excluded (included) anyway.
msoo [0]	controls the inclusion of the two-electron spin-other-orbit interaction in the same way as the **mso** parameter.
mss [0]	controls the inclusion of the spin–spin interaction in the same way as the **mso** parameter.
moo [0]	controls the inclusion of the orbit–orbit interaction in the same way as the **mso** parameter.

imvc [0]	indicates the mode for processing the mass-velocity term: `imvc=+1` include mass-velocity term directly in the Hamiltonian (7.5). `imvc=-1` include the mass-velocity term only for the bound orbitals. For scattering orbitals this interaction can then be included in the BSR_HD program as a first-order correct `imvc=-2` the same as `imvc=-1` but additionally exclude the mass-velocity term for bound orbitals with principal quantum number **nmvc**. This mode was introduced for consistency in cases when some bound orbitals were generated in previous bound-state BSR calculations without inclusion of relativistic corrections.
nmvc [0]	the principal quantum number of orbitals for which the mass-velocity term is not included (see the `imvc= -2` mode above).
mrecl [100000]	maximum record length (in bytes) in unformatted files. This parameter was introduced to insure the portability of the unformatted files on different computers and platforms.
nud [16]	unit number for the debug **int_mat.nnn** file. If **nud** $= 0$ no debug file will be created.
eps_c [1.d-10]	tolerance for angular coefficients combined with numerical overlap factor. Coefficients less than **eps_c** are ignored.
eps_det [1.d-10]	tolerance for the value of the numerical overlap factor. Overlap factors less than **eps_det** are taken as zero.
eps_soo [1.d-2]	tolerance for the angular coefficients for the two-electron Breit–Pauli terms. The treatment of these terms may take most of the time. On the other hand, they give only small corrections, and thus this parameter allows us to consider contributions only from the principal configurations.
nphys [0]	number of physical target states. If **nphys** $= 0$, all states are considered physical.
iphys [0]	mode for processing the non-physical (pseudo) target states. If **iphys** > 0, all interactions between pseudostates are ignored. If **iphys** $= 1$, the pseudostates interact only with the ground state.
maxnc [100000]	size (in bytes) of buffer for coefficients read from **int_bnk**. The buffer was introduced in order to reduce the number of read-write operations. It is especially important if several **bsr_mat** programs for different partial waves are run with a single disk.

7.5. Output of Hamiltonian matrices and asymptotic coefficients

The main result of the **bsr_mat** program is the generation of the interaction matrix (2.17) in the B-spline representation (7.5). The latter is stored in the **bsr_mat.nnn** files for each partial wave separately. Below is a summary of the output records in these files.

1. `ns,nch,kcp`
   `ns` – number of B-splines.
   `nch` – number of channels.
   `ncp` – number of bound $(N+1)$-electron configurations.
   The total dimension of interaction matrix is `nhm=ns` $\times$ `nch+ncp`.
2. `om`$(i, 1:i)$, repeat for each row $i = 1$,`nhm` – total overlap matrix.
3. `hm`$(i, 1:i)$, repeat for each row $i = 1$,`nhm` – total interaction matrix.
4. `mk` – maximum multipole index.
5. `CF(1:nch,1:nch,0:mk)` – matrix of asymptotic coefficients (2.26).
6. `t(ns+1)`
   Array `t(1:ns+ks)` is the set of knot points used for the generation of B-splines.
   Value `t(ns+1)` defines the maximum radius for the B-splines, i.e. the R-matrix radius.
7. `th(1:ntarg,1:ntarg)`
   `ntarg` – number of target states.
   `th` – interaction matrix between target states.

`th` is a by-product of the calculations of the total interaction matrix. This matrix should be diagonal with the target energies on the diagonal. Comparison with the input target energies allows us to control the accuracy of the calculations.

8. `to(1:ntarg,1:ntarg)`
   `to` – overlap matrix between target states. It should be equal to the unit matrix. A deviation from the unit matrix indicates that the input target file is not correct.

9. `EC` – energy (in a.u.) of the common closed shells.

8. Program BSR_HD

8.1. Structure and data flow

This stage of the BSR 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.

The block diagram of the program BSR_HD, along with the data flow, is given in Fig. 7. First, the program reads the data common to all partial waves. This includes the parameters of the calculation (routine **read_arg**), the B-spline parameters (**define_spline**), and the description of the target states **(read_target)**. Then the program executes fully independent calculations for each partial wave under consideration.

At this stage, the program first reads the interaction matrix from the corresponding file **bsr_mat.nnn**, checking the option to use experimental target energies instead of theoretical values. This option is controlled by the input parameter **iexp**. If this parameter is greater than 0, the program reads experimental threshold energies from the file **thresholds** and calls the routine **exp_thresh** for the corresponding modification of the interaction matrix. This modification is not as straightforward as in case of orthogonal orbitals; in general, it needs an additional diagonalization procedure. For example, in order to shift all eigenvalues by the same value E_c, we should add to the Hamiltonian matrix $\boldsymbol{H}$ the correction $E_c\boldsymbol{S}$, where $\boldsymbol{S}$ is the total overlap matrix. Depending on the orthogonality conditions imposed on the continuum orbitals, the total overlap matrix has a different structure. In case of orthogonal orbitals, or for high partial waves, when the orbital angular momenta of the continuum orbitals are greater than the momenta of the bound orbitals, the overlap matrix consists simply of diagonal blocks of the banded B-spline overlap matrix $\boldsymbol{B}$ (7.6). In this case, the experimental-thresholds correction consists simply of adding $(E_{\rm exp} - E_{\rm theor})\boldsymbol{B}$ to each diagonal block $H(ii)$ in matrix (7.5). This option is realized when **iexp** $= 1$. In case **iexp** $= 2$, the correction to the diagonal block $H(ii)$ is $(E_{\rm exp} - E_{\rm theor})S(i,i)$, where $S(i,i)$ is the corresponding diagonal block in the total overlap matrix. This correction can be applied in the case of restricted non-orthogonality, when the non-diagonal blocks in the total overlap matrix are small in comparison to the diagonal blocks. The

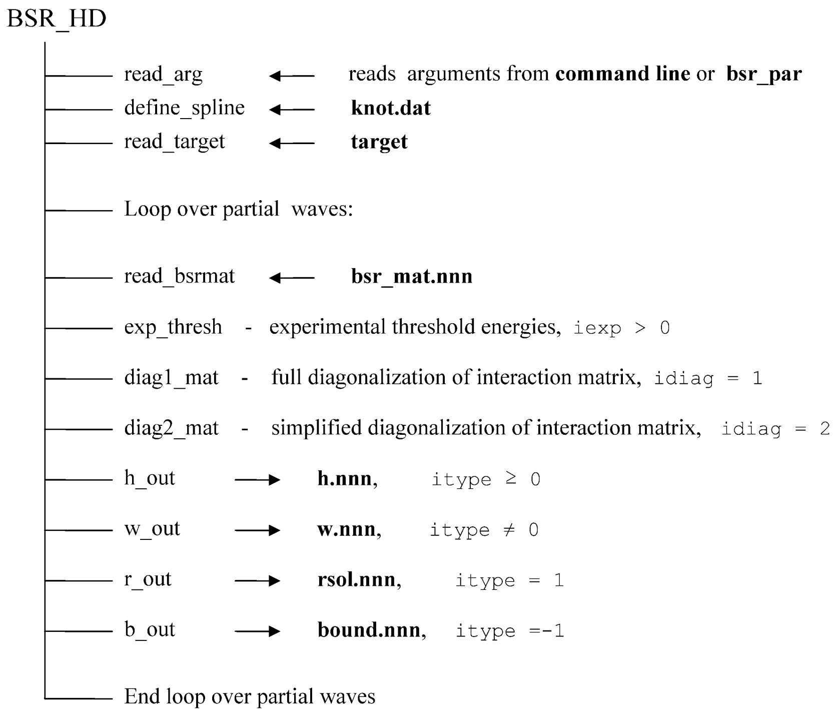
BSR_HD
read_arg
reads arguments from command line or bsr_par
define_spline
knot.dat
read_target
target
Loop over partial waves:
read_bsrmat
bsr_mat.nnn
exp_thresh - experimental threshold energies, iexp > 0
diag1_mat - full diagonalization of interaction matrix, idiag = 1
diag2_mat - simplified diagonalization of interaction matrix, idiag = 2
h_out
h.nnn, itype ≥ 0
w_out
w.nnn, itype ≠ 0
r_out
rsol.nnn, itype = 1
b_out
bound.nnn, itype =-1
End loop over partial waves

Fig. 7. Block diagram for the program BSR_HD and data flow (see text).

general procedure is applied if **iexp** = 3 and consists of the following steps. First, we introduce the orthogonal matrix $\boldsymbol{O}$ which diagonalizes $\boldsymbol{S}$ according to

$$\boldsymbol{O}^{\mathrm{T}} \cdot \boldsymbol{S} \cdot \boldsymbol{O} = \boldsymbol{s}, \tag{8.1}$$

where $\boldsymbol{s}$ is diagonal. We now define the square root matrix $\boldsymbol{S}^{1/2}$ as

$$\boldsymbol{S}^{1/2} = \boldsymbol{O} \cdot \boldsymbol{s}^{1/2} \cdot \boldsymbol{O}^{\mathrm{T}}. \tag{8.2}$$

Then the eigenproblem (7.1) can be written as

$$\boldsymbol{S}^{-1/2}\boldsymbol{H}\boldsymbol{S}^{-1/2}(\boldsymbol{S}^{1/2}\boldsymbol{C}) = (\boldsymbol{S}^{1/2}\boldsymbol{C})\boldsymbol{E}, \tag{8.3}$$

where $\boldsymbol{E}$ is the diagonal matrix of eigenvalues and $\boldsymbol{C}$ is the matrix of solutions. Now we can apply the usual procedure for adjusting the target energies, i.e. add to each diagonal element of the modified Hamiltonian matrix the corresponding value of $(E_{\mathrm{exp}} - E_{\mathrm{theor}})$. If we denote these corrections by the diagonal matrix $\boldsymbol{D}$, we have

$$(\boldsymbol{S}^{-1/2}\boldsymbol{H}\boldsymbol{S}^{-1/2} + \boldsymbol{D})(\boldsymbol{S}^{1/2}\boldsymbol{C}) = (\boldsymbol{S}^{1/2}\boldsymbol{C})\boldsymbol{E}. \tag{8.4}$$

Returning now to the original notation, we obtain

$$(\boldsymbol{H} + \boldsymbol{S}^{1/2}\boldsymbol{D}\boldsymbol{S}^{1/2})\boldsymbol{C} = \boldsymbol{S}\boldsymbol{C}\boldsymbol{E}, \tag{8.5}$$

i.e. the final correction is defined by the second term in the brackets. The above general procedure requires an additional diagonalization of the overlap matrix and may be time-consuming in large-scale cases. Hence, it is recommended to apply this option only for low partial waves where we expect a large contribution to the interaction matrix due to one-electron overlaps (see Table 1).

The next step is the diagonalization of the interaction matrix. It can be done in the full or simplified mode that is defined by the input parameter **idiag**. In full diagonalization mode, **idiag** = 1, the only further modification of the interaction matrix is due to the boundary conditions. Since the first B-spline is the only spline with nonzero value at $r = 0$, the zero boundary condition at $r = 0$ is imposed by removing the first B-spline from all expansions. In the code it means that we remove in the Hamiltonian matrix all rows and columns related to the first spline. A further modification is related to the known dependence proportional to r^{l+1} for the one-electron radial functions at small r. Recall that the first k_s splines $B_i^{k_s}$ behave as r^{i-1} at small r. Hence we can guarantee the r^{l+1} behavior of the one-electron radial functions simply by removing the first $(l+1)$ splines from expansions. This can be done only for orbitals with angular momentum $l \leqslant k_s - 2$. For the remaining orbitals the r^{l+1} behavior is only approximately reproduced by a suitable combination of the first splines. In bound-state calculations with the input parameter **itype** = −1, we additionally remove the last two splines from the B-spline expansion, in order to guarantee that the resulting solutions have zero values and zero first derivates at the border $r = a$.

When we diagonalize the Hamiltonian matrix in the B-spline basis, we get a very wide spectrum of eigenvalues, which spread from threshold energies up to positive energies on the absolute energy scale. This is related to the effective completeness of the B-spline basis, and hence no Buttle correction is required in this case. On the other hand, the B-spline expansions are comparatively large, and in practical calculations it is not reasonable to treat all eigenvalues on equal footing. For the high-lying solutions, which provide only a small contribution to the final R-matrix, we may use an approximate treatment. One of such treatments is realized in the routine **diag2_mat**, which is called when the input parameter **idiag** = 2. In this case, we first diagonalize only the diagonal blocks in the matrix (6.5) which are connected to the given scattering channels. Then the interaction matrix is transformed to this new basis, and in the final expansion we retain only those solutions with energies below some value **Emax**. This parameter is an input parameter and defined by the user on the basis of the physical model under consideration. Then the new simplified matrix is diagonalized, and the resulting solutions are very close to the solution of the full matrix for energies below **Emax**. The remaining solutions are approximated by those obtained at the diagonalization of the diagonal blocks. They can be considered as some kind of Buttle correction. Such simplified treatment can reduce the size of the interaction matrix by up to factor of 2, and hence considerably reduce the computational time.

The two-step diagonalization of the interaction matrix also allows us to introduce additional corrections at the level of first-order perturbation theory. When we have zero-order solutions, $\boldsymbol{c}_i$, obtained from the diagonalization of the diagonal blocks, we can add to the Hamiltonian matrix any perturbation as $\langle \boldsymbol{c}_{i'}|V|\boldsymbol{c}_i\rangle$, where V is the perturbation potential. Such a possibility is realized in the present implementation in the case of the mass-velocity correction for scattering orbitals, which is controlled by the parameter **jmvc**. Note that the direct inclusion of the mass-velocity operator (2.41) in the Hamiltonian matrix leads to unphysical solutions with very low energies.

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 corresponding H file in the routine **h_out**. The output H file has a standard structure, first adopted in the Belfast R-matrix code. 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 simply 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. Another important difference from the earlier codes is that we do not provide any Buttle correction in the H file. The user should remember that the resulting H file can only be used with the parameter **ibuttle** $= 0$ in further calculations, for example, when running the FARM program.

In photoionization calculations, the program also records the full set of inner-region solutions in file **rsol.nnn**. These data are then used by program BSR_DMAT to generate the corresponding dipole matrix.

The last option in the BSR_HD program is a bound-state calculation. The resulting bound-state solutions are generated by the routine **b_out**. In general, we can get a huge number of solutions, not all of which are physical or even helpful in further calculations. A set of input parameters is therefore introduced in order to avoid the generation of a large number of output **bound.nnn** files with excess information. The parameter **msol** limits the total number of output solutions, the parameters **Emin** and **Emax** restrict the solutions according to their energies, the parameter **it_max** provides the highest target threshold for output solutions, and, finally, the parameters **n_min** and **n_max** restrict the output solutions according to their principal quantum numbers. The principal quantum numbers for bound-state solutions are defined according to their binding energies,

$$E_b = \frac{z^2}{(n-\mu)^2}. \tag{8.6}$$

For a more accurate determination of the n values, the user may provide specific quantum defects μ_l through the parameters **qd_0, qd_1, qd_2** and **qd_3** for orbital angular momenta $l = 0, 1, 2, 3$, respectively.

The routine **b_out** also tries to provide the spectroscopic assignment for the output solutions. To do this, the weights of different channels and configurations are needed. Since the solutions vectors (7.2) have the metric

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{c} = 1, \tag{8.7}$$

the problem of weight determination is not trivial in the present case. For the determination of the weights we use the vector $\boldsymbol{S}^{1/2}\boldsymbol{c}$, where $\boldsymbol{S}^{1/2}$ is defined in (8.2). It again requires the diagonalization of the total overlap matrix. 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.

8.2. Data files

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

bsr_par	File type: formatted sequential input. Written by user. Read by routine **read_arg**. Description: input parameters for given run.
target	File type: formatted sequential input. Written by user and modified by BSR_PREP and BSR_CONF programs. Read by routine **read_target**. Description: contains the description of the target states and scattering channels.
knot.dat	File type: formatted sequential input. Written by user. Read and modified by routine **define_grid** from the BSPLINE library. Description: input parameters that define the B-spline grid.
cfg.nnn	File type: formatted sequential input. Created by program BSR_CONF. Read by routine **read_conf**. Description: list of configurations in spectroscopic notation for the partial wave **nnn**.
target.bsw	File type: unformatted sequential input. Created by program BSR_PREP. Read by routine **read_bsw**. Description: target one-electron orbitals in the B-spline basis.

bsr_mat.nnn	File type: unformatted sequential input. Created by program BSR_MAT. Read by routine **read_bsrmat**. Description: Hamiltonian matrix and asymptotic coefficients.
thresholds	File type: formatted sequential input. Written by user. Read by routine **read_arg**. Description: experiment target energies, optional.
bsr_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, STG4, BSR_PHOT or other programs that deal with the outer region. Description: diagonalized Hamiltonian matrix data for given partial wave.
bound.nnn	File type: formatted sequential output. Created by routine **b_out**. Read by program BSR_DMAT and BSR_MCHF. Description: bound-state solutions for given partial wave in the B-spline close-coupling representation (7.2).
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.

8.3. Input data

Input data can be provided in the command line or in the input file **bsr_par** (the data from the command line overwrite the data from the input file). Below we describe those data from **bsr_par** which are read by the program **bsr_hd**. 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 **bsr_hd**.

klsp1 [1]	first partial wave under consideration.
klsp2 [klsp1]	last partial wave under consideration.
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; **bound.nnn** file is generated.
iexp [0]	controls the treatment of experimental thresholds.
idiag [1]	diagonalization mode: `idiag=1` – full diagonalization of interaction matrix; routine **diag1_mat**. `idiag=2` – simplified diagonalization; routine **diag2_mat**.
jmvc [0]	pointer to the inclusion of the mass-velocity term as a first-order perturbation.

Parameters for bound-state calculations:

msol [0]	restricts the number of bound-state solutions for output in **bound.nnn** file. `msol=0` means 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).
it_max [1]	the highest target threshold for bound-state calculations.
n_min [1]	the minimum principal quantum number of bound-state solutions.
n_max [99]	the maximum principal quantum number of bound-state solutions.
qd_0 [3.0]	quantum defect for s-orbitals.
qd_1 [2.5]	quantum defect for p-orbitals.
qd_2 [2.0]	quantum defect for d-orbitals.
qd_3 [0.5]	quantum defect for f-orbitals.

The above values for quantum defects are used to classify the bound solutions.

8.4. Structure of the H file

We keep the same structure for the H files as in the Belfast R-matrix code [4]. The **h.nnn** files from the BSR_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.

10. `nelc,nz,lrange,km,ntarg,RA,BSTO`

    `nelc` – number of electrons in the target.
    `nz` – atomic number.
    `lrange` – maximum value of $(l+1)$ for the continuum orbitals.
    `km` – maximum multipole index for the asymptotic coefficients.
    `RA` – R-matrix radius.
    `BSTO {0.0}` – the value of the logarithmic derivative to be imposed on the continuum orbitals.
    The variables lrange and BSTO were included only for consistency with earlier versions.

11. `etarg (1:ntarg)`
12. `ltarg (1:ntarg)`
13. `starg (1:ntarg),ptarg(1:ntarg)`

    `etarg` – target energies, in a.u.
    `ltarg` – total L for target states ($2J$ in case of intermediate coupling).
    `starg` – total S for target states (0 in case of intermediate coupling).
    `ptarg` – parity of the target states.

14. `(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.

15. `lpar,spar,ipar,nch,nhm,more`

    `lpar` – total L for given partial wave ($2J$ in case of intermediate coupling).
    `spar` – total S for given partial wave (0 in case of intermediate coupling).
    `ipar` – total parity for given partial wave.
    `nch` – number of scattering channels.
    `nhm` – number of inner-region solutions.
    `more {0}` – indicates no other information, except for the given partial wave.

16. `NCONAT(1:ntarg)` – indicates the number of channels associated with each target state.
17. `lch(1:nch)` – angular momenta of channel orbitals.
18. `CF(1:nch,1:nch,1:mk)` – matrix of asymptotic coefficients (2.26).
19. `eval(1:nhm)` – R-matrix poles in descending order (values E_k in Eq. (2.20)).
20. `wmat(1:nch,1:nhm)` – corresponding surface amplitudes (values w_{ik} in Eq. (2.20)).

8.5. Output of the R-matrix solutions

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

1. `nhm,khm,kch,kcp,ns`
2. `eval(1:khm)`
3. `a(1:nhm,:)`
   repeat 3 `khm` times

`nhm` – dimension of the Hamiltonian matrix.
`khm` – number of solutions.
`kch` – number of channels.
`kcp` – number of $(N+1)$ configurations.
`ns` – number of B-splines for each channel.
`eval(.)` – R-matrix eigenvalues.
`a(1:nhm,.)` – corresponding eigenvectors.

8.6. Output of bound solutions

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

1. `ns,nch,ncp,nhm,nbound,L,S,P`
2. `i,LABEL`
3. `eval(i),Ebind(i),eff_n(i)`
4. `a(1:nhm,i)`
   repeat records 2–4 for each bound state.

`ns` – number of B-splines.
`nch` – number of channels.
`ncp` – number of $(N+1)$-electron configurations (perturber).
`nhm` – dimension of solution vector in the B-spline representation.
`LABEL` – spectroscopic notation for solution i.
`eval` – energy in a.u.
`Ebind` – binding energy.
`eff_n` – effective principal number.
`a` – solution vector in the B-spline representation, see Eq. (7.2).

8.7. Output of weights

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

1. `nch, ncp, khm`
2. `W(1:nch+ncp)`
   repeat record 2 for each solution

`nch` – number of channels.
`ncp` – number of $(N+1)$-electron configurations (perturber).
`khm` – number of solutions.
`W` – array of weights for each channel and $(N+1)$-electron configuration.

9. Program MULT

9.1. Outline of the MULT calculations

The MULT program performs the angular integrations necessary to express the matrix elements of the multipole transition operators as a linear combination of radial integrals. Any amount of non-orthogonality between the orbitals may be present, leading to overlap factors in the matrix elements. The program can effectively reuse data obtained previously by creating a databank for angular coefficients. MULT is a new, completely revised and optimized version of the general program ZAP_NO [24] and can be used in many other aspects of atomic-structure calculations. MULT can also be considered as a straightforward extension of the program MCHF_MLTPOL [56] to the case of arbitrary orthogonal conditions between orbitals in the initial and final states. (In the following write-up, we try to keep the same notation as in MCHF_MLTPOL.) The calculations of the angular integrals follow the method based upon the representation of configuration wavefunctions through Slater determinants. The details of the present approach can be found in the long write-up of the program ZAP_NO. Below we only provide a brief description of the present version, outlining the main modifications made in comparison to ZAP_NO.

9.1.1. Definitions

The program deals with the transition operators $O_\mu^{[\lambda]}$ for electric and magnetic transitions, which can be represented in tensor forms, respectively, as

$$E_\mu^{[\lambda]} = e \sum_i r^\lambda(i) C_\mu^{[\lambda]}(i) \tag{9.1}$$

and

$$M_\mu^{[\lambda]} = 2\beta\sqrt{\lambda(2\lambda-1)} \sum_i \left\{ r^{\lambda-1}(i) C^{[\lambda-1]}(i) \times \left[\frac{1}{\lambda+1} l^{[1]}(i) + \frac{1}{2} g_s s^{[1]}(i) \right] \right\}_\mu^{[\lambda]}. \tag{9.2}$$

Here $\beta = e\hbar/2mc$ is the Bohr magneton, g_s is the electron g-factor, and the summation is over the atomic electrons. In the Breit–Pauli approximation, the initial-state and final-state wavefunctions are superpositions of LS-configuration state functions, i.e.

$$\Psi(JM) = \sum_i c_i \Phi(\gamma_i L_i S_i JM). \tag{9.3}$$

The dependence of the transition matrix elements on the magnetic quantum numbers can be found by applying the Wigner–Eckart theorem,

$$\langle JM|O_\mu^{[\lambda]}|J'M'\rangle = (-1)^{J-M} \begin{pmatrix} J & \lambda & J' \\ -M & \mu & M' \end{pmatrix} \langle J\|O^{[\lambda]}\|J'\rangle, \tag{9.4}$$

and the reduced matrix element (RME) $\langle J\|O^{[\lambda]}\|J'\rangle$ itself can be written as

$$\langle J\|O^{[\lambda]}\|J'\rangle = \sum_{ii'} c_i c_{i'} \langle \gamma_i L_i S_i J\|O^{[\lambda]}\|\gamma_{i'} L_{i'} S_{i'} J'\rangle. \tag{9.5}$$

The operator $E_\mu^{[\lambda]}$ in (9.1) associated with the *electric* transition Eλ is completely spin-independent, and we can use the "uncoupling" formula for its reduced matrix elements

$$\langle \gamma LSJ\|E^{[\lambda]}\|\gamma' L'S'J'\rangle = \delta_{SS'}(-1)^{L+S+J'+\lambda}[J,J']^{1/2} \begin{Bmatrix} L & S & J \\ J' & \lambda & L' \end{Bmatrix} \langle \gamma LS\|E^{[\lambda]}\|\gamma' L'S'\rangle. \tag{9.6}$$

The treatment of magnetic multipole transitions of any order is more complex. Usually, the magnetic operator (9.2) is split into two parts with different spin and spatial dependence:

$$M_\mu^{[\lambda]} = 2\beta\sqrt{\lambda(2\lambda-1)} \left[\frac{1}{\lambda+1} \mathrm{MA}_\mu^{[\lambda]} + \frac{1}{2} g_s \mathrm{MB}_\mu^{[\lambda]} \right], \tag{9.7}$$

with

$$\mathrm{MA}_\mu^{[\lambda]} = \sum_i r^{\lambda-1}(i) \left\{ C^{[\lambda-1]}(i) \times l^{[1]}(i) \right\}_\mu^{[\lambda]} \tag{9.8}$$

and

$$\mathrm{MB}_\mu^{[\lambda]} = \sum_i r^{\lambda-1}(i) \left\{ C^{[\lambda-1]}(i) \times s^{[1]}(i) \right\}_\mu^{[\lambda]}. \tag{9.9}$$

The uncoupling formulae are different for these parts:

$$\langle \gamma LSJ\|\mathrm{MA}^{[\lambda]}\|\gamma' L'S'J'\rangle = \delta_{SS'}(-1)^{L+S+J'+\lambda}[J,J']^{1/2} \begin{Bmatrix} L & S & J \\ J' & \lambda & L' \end{Bmatrix} \langle \gamma LS\|\mathrm{MA}^{[\lambda]}\|\gamma' L'S'\rangle \tag{9.10}$$

and

$$\langle \gamma LSJ\|\mathrm{MB}^{[\lambda]}\|\gamma' L'S'J'\rangle = [J,J',\lambda]^{1/2} \begin{Bmatrix} L & S & J \\ L' & S' & J' \\ \lambda-1 & 1 & \lambda \end{Bmatrix} \langle \gamma LS\|\mathrm{MB}^{[\lambda]}\|\gamma' L'S'\rangle. \tag{9.11}$$

The remaining problem is to evaluate the RME of the electric and magnetic multipole operator of any order between arbitrary *LS*-coupled configurations. This is done in the present programs based upon the representation of configuration wavefunctions through Slater determinants. This allows us to introduce a completely unrestricted version of a non-orthogonal scheme.

9.1.2. Calculation of RME between LS configuration wavefunctions

The present procedure is based upon the expansion of the N-electron configuration wave function in terms of Slater determinants

$$\Phi(\gamma LSM_LM_S) = \sum_{\{u\}} \big(\{u\}|\gamma LS\big)|u_1 \ldots u_N\rangle, \tag{9.12}$$

where $(\{u\}|\gamma LS)$ denotes the expansion coefficient, and the individual determinant $|u_1 \ldots u_N\rangle$ has the form

$$|u_1 \ldots u_N\rangle = \frac{1}{N!}\begin{vmatrix} \langle t_1|u_1\rangle & \langle t_1|u_2\rangle & \ldots & \langle t_1|u_N\rangle \\ \langle t_2|u_1\rangle & \langle t_2|u_2\rangle & \ldots & \langle t_2|u_N\rangle \\ \ldots & \ldots & \ldots & \ldots \\ \langle t_N|u_1\rangle & \langle t_N|u_2\rangle & \ldots & \langle t_N|u_N\rangle \end{vmatrix}. \tag{9.13}$$

Here $t = (\boldsymbol{r}, s)$ represents the spin and position of the individual electrons and u stands for the one-electron quantum numbers, $|u\rangle = |nlm\mu\rangle$. Thus, the determination of matrix elements between CSFs is reduced to that of matrix elements between separate Slater determinants, which, in turn, are reduced to one-electron integrals between spin–orbitals in the $nlm\mu$-representation. In comparison to the existing codes based upon the Racah technique, the resulting formulae explicitly include the magnetic quantum numbers and contain additional summation over Slater functions and over all electrons but not shells. This increases considerably the number of operations required, and the speed of the calculations is much reduced. On the other hand, the final formulae are much simpler and more convenient for programming. The principal advantage, however, is that the $nlm\mu$-representation procedure facilitates the straightforward extension to the case of non-orthogonal orbitals to be made in the most general way. The main problem here is to obtain the determinantal representation of arbitrary atomic wavefunctions, including configurations with several open shells. An effective way to solve this problem has been proposed in the program ZAP_NO. It is based upon the preliminary creation of tables of expansion coefficients for the individual shells. The determination of the coefficients for the CSFs is then reduced to the vector coupling of shell terms. This procedure is used in the present code. For details we refer the reader to the description of the program ZAP_NO.

When we get expansions (9.12) for the initial and final *LS* states, the total matrix elements for transition operators are reduced to matrix elements between Slater determinants. Consider a matrix element between two Slater determinants, $|u_1 \ldots u_N\rangle$ and $|v_1 \ldots v_N\rangle$, for a transition operator $\boldsymbol{O}^k = \sum_{i=1}^N \boldsymbol{o}^\lambda(i)$, where $\boldsymbol{o}^\lambda(i)$ is the one-electron operator with multipolarity λ, acting on electron i. Let us define the determinant $D_{uv} = \det\{\langle u_i|v_j\rangle\}$ of the matrix formed from all overlap integrals between the one-electron orbitals u_i and v_j $(i.j = 1, \ldots, N)$ and let the $D_{uv}(\rho, \rho')$ denote the first cofactors of the determinant D_{uv}, corresponding to the matrix of overlap integrals with orbitals u_ρ and v'_ρ deleted from the initial and final sets, respectively. Then, it can be shown [57] that

$$\langle u_1u_2 \ldots u_N|O_\mu^{[\lambda]}|v_1v_2 \ldots v_N\rangle = \sum_{\rho,\rho'} (-1)^{\rho+\rho'}\langle u_\rho|o^{[\lambda]}|v_{\rho'}\rangle D_{uv}(\rho, \rho'). \tag{9.14}$$

This formula, in connection with irreducible tensor algebra, allows us to reduce the configuration matrix elements to one-electron matrix elements between spin–orbitals $nlm\mu$, multiplied by overlap factors in the case of non-orthogonal radial orbitals. Applying the Wigner–Eckart theorem (9.4), the one-electron matrix elements in (9.14) can be reduced to one-electron RMEs. Here, however, we should consider two cases, one related to the electric and MA operators, and one to the MB operator. In the first case, we have no spin-dependence, and hence

$$\langle nlmm_s|o_\mu^{[\lambda]}|n'l'm'm'_s\rangle = (-1)^{l-m}\begin{pmatrix} l & \lambda & l' \\ -m & \mu & m' \end{pmatrix}\langle nl\|O^{[\lambda]}\|nl'\rangle\delta(m_s, m'_s). \tag{9.15}$$

For the MB operator (9.9), on the other hand, we should apply the formula for the matrix element of an irreducible tensor product [58] of two operators, which only depend on the variables of the first and second subsystem, respectively:

$$\begin{aligned}\langle nlmm_s|mb_\mu^{[\lambda]}|n'l'm'm'_s\rangle = (-1)^{l-m}&\begin{pmatrix} l & \lambda-1 & l' \\ -m & \mu_l & m' \end{pmatrix}(-1)^{1/2-m_s}\begin{pmatrix} \frac{1}{2} & 1 & \frac{1}{2} \\ -m_s & \mu_s & m'_s \end{pmatrix} \\ &\times C(\lambda-1, \mu_l, 1, \mu_s; \lambda, \mu)\langle nl\|mb^{[\lambda]}\|nl'\rangle,\end{aligned} \tag{9.16}$$

where C denotes a Clebsch–Gordon coefficient. Furthermore, μ, μ_l and μ_s are magnetic numbers defined by the total magnetic numbers of the configurations under consideration: $\mu_l = M_L - M'_L, \mu_l = M_S - M'_S, \mu = \mu_l + \mu_s$. In our case of electric and magnetic transitions (9.1) and (9.2), the one-electron RME can be of three different types:

$$\langle i\|e^{[\lambda]}\|j\rangle = \langle i\|r^\lambda C^{[\lambda]}\|j\rangle = (l_i\|C^{[\lambda]}\|l_j)R_{ij}^\lambda, \tag{9.17}$$

$$\begin{aligned}\langle i\|ma^{[\lambda]}\|j\rangle &= \langle i\|r^{\lambda-1}\{C^{[\lambda-1]}\times l^{[1]}\}^{[\lambda]}\|j\rangle\\ &= (-1)^{l_i+\lambda+l_j}\sqrt{l_j(l_j+1)(2l_j+1)(2\lambda+1)}\begin{Bmatrix}\lambda-1 & 1 & \lambda\\ l_j & l_i & l_j\end{Bmatrix}(l_i\|C^{[\lambda]}\|l_j)R_{ij}^{\lambda-1},\end{aligned} \tag{9.18}$$

and

$$\langle i\|mb^{[\lambda]}\|j\rangle = \langle i\|r^{\lambda-1}C^{[\lambda-1]}s^{[1]}\|j\rangle = \sqrt{\frac{3}{2}}(l_i\|C^{[\lambda-1]}\|l_j)R_{ij}^{\lambda-1}, \tag{9.19}$$

where R_{ij}^{λ} is the radial transition integral

$$R_{ij}^{\lambda} = \int_0^{\infty} \mathrm{d}r\, r^{\lambda} P_{n_i l_i}(r) P_{n_j l_j}(r). \tag{9.20}$$

The present algorithm for the calculation of matrix elements consists of two independent steps: the determination of the determinantal representation for two input configuration wavefunctions with certain magnetic quantum numbers M_L and M_S, and the calculation of matrix elements between all Slater states obtained by the above formulae. Our final task is to obtain the reduced matrix elements for configuration wavefunctions, $\langle\gamma LS\| O^{[\lambda]}\|\gamma' L'S'\rangle$. They are obtained on the basis of the calculated matrix elements $\langle\gamma LSM_LM_S\|E_{\mu}^{[\lambda]}\|\gamma' L'S'J'M_L'M_S'\rangle$ by a reciprocal application of the Wigner–Eckart theorem. The corresponding formulae can be obtained from Eqs. (9.15) and (9.16) by a straightforward replacing of the one-electron quantum numbers by the total ones. Note that the amount of calculations in the present approach crucially depends on the number of Slater states involved, which, in turn, depends on the chosen quantum numbers M_L and M_S. In the present implementation, we use $M_L = L$ and $M_S = S$, which guarantees a minimum number of relevant Slater states.

9.1.3. Output results

The MULT program provides, as a final result, the expressions for the reduced matrix elements of the electric and magnetic operators (9.1) and (9.2) in the form

$$\big\langle \Phi\big(\{nl\}\gamma LS\big)\big\|O\big\|\Phi'\big(\{n'l'\}\gamma' L'S'\big)\big\rangle = \sum a_k d^k(i,i')\times D\big(\{nl\},\{n'l'\}\big), \tag{9.21}$$

where a_k are the numeric coefficients which depend only on the angular symmetry of the configuration state functions involved, d^k is the corresponding radial integral dependent on the operator **O** under consideration, and $D(\{nl\},\{n'l'\})$ is the overlap factor, which depends only on the sets of radial orbitals used in constructing the wavefunctions Φ and Φ', respectively. It should be emphasized that the coefficients a_k do not depend directly on the set of principal quantum numbers $\{n\}$ and $\{n'\}$. They only depend on the orthogonality conditions imposed on the one-electron radial functions. For example, in the limiting case of orthogonal radial orbitals, the overlap factors are reduced trivially to factors 1 or 0, and we obtain the weighted sum of radial integrals d^k, which is ordinarily used in atomic structure calculations. In the case of intermediate orthogonality conditions, some D-factors disappear while others are simplified. Thus, once we have obtained the matrix element in the case of fully non-orthogonal orbitals, we can also obtain the matrix elements for all other configurations with the same angular symmetry and for all other orthogonal conditions, only by analyzing the overlap factors $D(\{nl\},\{n'l'\})$ and replacing the set $\{n,n'\}$ by the specific one. Such a scheme is realized in the present code. In large-scale calculations, this can lead to a considerable reduction of execution time, because in practical calculations the set of configurations contains many configurations with the same angular symmetry.

On the other hand, rather than record the angular coefficients for the given set of CSFs, we can save only the expansions for the matrix elements with different angular symmetries, supposing that all radial functions in the initial and final sets are non-orthogonal. These data may be considered as a *databank* for angular coefficients. They allow us to obtain the matrix element expansions for any set of CSFs constructed with the angular symmetries included in the bank. For new symmetries, we only have to perform the additional calculations and the databank can be extended.

9.2. Structure and data flow

The block diagram of the program MULT, along with the data flow, is given in Fig. 8. The only input data files are c-files with a list of configuration state functions for the initial and final states involved. The MULT program considers only one type of transition operator in a single run. The type of the operator is given as E1, E2, ..., M1, M2, ..., where the first letter indicates if it is an electric or magnetic transition while the second letter defines the multipole index. The parameter E0 indicates the calculations of overlap matrix elements. Since MULT only requires a small number of input parameters, all of them must be defined in the command line as arguments.

The **r_conf** routine reads the input configuration lists in spectroscopic notation, decodes them to the internal 'integer' representation, and tests the CSFs as to the correctness of angular coupling, number of electrons, parity and AFTER conditions. **r_conf** also allocates (or reallocates) all main arrays with dimensions specific for the given partial wave. Then the CSFs are sorted according

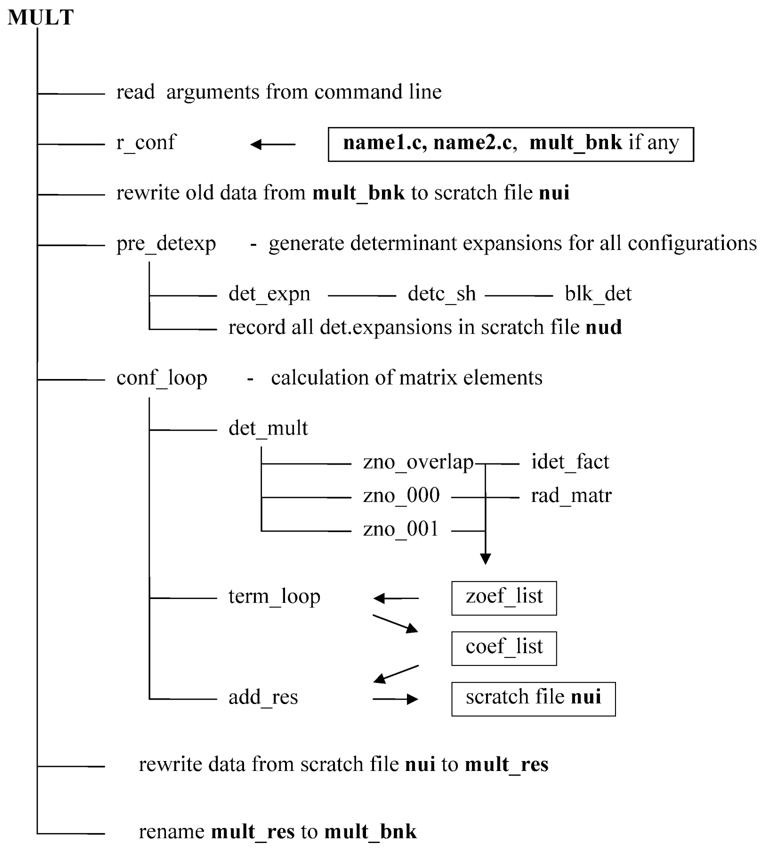
MULT
read arguments from command line
r_conf
name1.c, name2.c, mult_bnk if any
rewrite old data from mult_bnk to scratch file nui
pre_detexp - generate determinant expansions for all configurations
det_expn
detc_sh
blk_det
record all det.expansions in scratch file nud
conf_loop - calculation of matrix elements
det_mult
zno_overlap
idet_fact
zno_000
rad_matr
zno_001
term_loop
zoef_list
coef_list
add_res
scratch file nui
rewrite data from scratch file nui to mult_res
rename mult_res to mult_bnk

Fig. 8. Block diagram for the program MULT and data flow (see text).

to their angular and configuration symmetries. At this stage the program also checks if this is a continuing calculation, i.e. whether there exists a relevant databank from some previous calculations. The program analyses the information contained in the databank and defines the subset of configurations that need additional consideration, or it decides that the information in the databank is sufficient for the given input configuration lists.

In the next block, **pre_detexp**, the determinant expansions for all configurations are generated and recorded in the scratch file. This block is the main modification made in comparison to the previous version, ZAP_NO [24]. It avoids the large amount of recalculations of repeating determinant coefficients. The determinant expansion for a given angular symmetry is generated in the routine **det_expn**, on the basis of angular coupling coefficients, calculated in **detc_sh**, and the coefficients for determinant expansions of individual shells, stored in the blockdata **blk_det**.

The calculations of the full set of integral coefficients and overlap factors for the input angular symmetries are run in the block **conf_loop**. First, the routine **det_mult** computes the radial expansions for matrix elements between all Slater determinants involved. This information is accumulated in the module **zoef_list**. The cases of overlaps, zero-orbital or one-orbital difference in the Slater determinants under consideration are treated separately by the routines **zno_overlap**, **zno_000**, and **zno_001**, respectively. The routine **rad_matr** contains expressions for one-electron matrix elements in the $nlm\mu$-representation (see Section 9.1.2). The function **idet_fact** defines the corresponding overlaps factors $D_{uv}(\rho, \rho')$ in the expression (9.13).

Then, in the routine **term_loop**, the angular coefficients between two given angular symmetries are calculated on the basis of angular coefficients between the individual determinants involved (stored in module **zoef_list)** and determinant expansion coefficients, which are read from the scratch file **nud** (see block **pre_detesp)**. These data are accumulated in the module **coef_list**. When all determinants for the given angular symmetries are exhausted, the final angular coefficients are recorded by the routine **add_res** in the scratch file **nui**, and module **coef_list** is initialized to accumulate data from other matrix elements. The above calculations are repeated in the block **conf_loop** for all combinations of input angular symmetries.

Finally, the new data and data from previous calculations are stored in the file **mult_res**. This file then is renamed to the initial name **mult_bnk**. This procedure prevents loosing the data in the initial databank if an error occurs when running the MULT program. The format of the output databank is given in Section 9.5.

All calculations in MULT are carried out in full non-orthogonal mode, without specifying the principal quantum numbers for the orbitals involved. Instead, the positions of the corresponding shells in the configurations are used to specify the one-electron orbitals involved. Hence, the data obtained can be used further for all atomic states with the given angular symmetries. In the present version we also abandoned the creation of an additional list of angular coefficients for a specific set of one-electron orbitals. Instead, the matrix elements for specific orbitals are calculated dynamically when the required transition data are generated.

9.3. Data files

name1.c File type: formatted sequential input.
Created by user.
Description: contains configuration expansion for the initial state.

name2.c File type: formatted sequential input.
Created by user.
Description: contains configuration expansion for the final state.

mult_bnk File type: unformatted sequential output.
Written by program MULT.
Read by BRS_DMAT.
Description: databank for angular coefficients of the dipole operator.

mult.log File type: formatted sequential output.
Written by program MULT.
Read by user.
Description: running information.

9.4. Input parameters

Input parameters must be provided in the command line.

name1.c name for *c*-file of initial state.

name2.c name for *c*-file of final state.

AA character*2, type of operator and multipole index: E1, E2, ... , M1, M2, ...

9.5. Structure of the dipole databank

Summary of the output records in the **mult_bnk** file.

1. `ktype,kpol`

   `ktype` – ‘E’ or ‘M’, for electric and magnetic transitions, respectively.
   `kpol` – multipole index.

16. `nsymt,nsymc`

   `nsymt` – number of different angular symmetries.
   `nsymc` – number of different configuration symmetries.

17. `ASYM(1:nsymc)`

   List of configuration symmetries (character*26). Each configuration symmetry `ASYM(.)` contains `(l(i),q(i),i=1,no),` `LS` in format `(8(a1,i2),2a1)`, where `no` – number of shells; `l(i)` – electron orbital momentum in `i`th shell; `q(i)` – number of electrons in `i`th shell; `LS` – total term in spectroscopic notation.

18. `BSYM(1:nsymt)`

   List of angular symmetries (character*40). Each angular symmetry `BSYM(.)` contains a list of the generalized terms of shells `(A,L,S,LI,SI,1:no)` in format `(8(i1,a1,2i1,a1))`. Each generalized term includes: `A` – seniority number; `L` – orbital momentum of the shell; `S` – spin; `LI` – intermediate orbital momentum; `SI` – intermediate spin.

19. `IT_CONF(1:nsymt)`

   Indicates the corresponding configuration symmetry for a given angular symmetry, i.e. it defines the relation between the `ASYM` and `BSYM` arrays.

20. `IT_OPER(1:nsymt` × `(nsymt+1)/2)`

    `IT_OPER(ij)` indicates the operators, which have been already considered by the program for angular symmetries `i,j`, such that `ij = i` × `(i+1)/2+j, i > j.`

21. `ndet,kdet`
22. `KPD(1:ndet)`
23. `IPD(1:ndet)`
24. `NPD(1:kdet)`

    `ndet` – total number of overlap determinants.
    `kdet` – size of the `NPD` array, containing the description of all overlap determinants.
    `KPD(i)` – dimension of the `i`th overlap determinant, `kd`.
    `IPD(i)` – pointer on the `i`th overlap determinant in array `NPD`, `ip`.
    `NPD(ip+1:ip+kd)` – contains description of the ith determinant as `NPD(.) =` i_1`*`b_d `+` i_2, where $b_d = 2^{15}$ – packing basis for overlap determinants; i_1, i_2 – pointers to shells in the involved configurations.

25. `ndef,kdef`
26. `KPF(1:ndef)`
27. `IPF(1:ndef)`
28. `NPF(1:kdef)`

    `ndef` – number of different overlap factors.
    `kdef` – size of the `NPF` array, containing the description of all overlap factors.
    `KPF(i)` – number of determinants in the `i`th overlap factor, `kd`.
    `IPF(i)` – pointer on the `i`th overlap factor in array `NPF`, `ip`.
    `NPF(ip+1:ip+kd)` – contains description of the `i`th overlap factor as `NPF(.) = ipd*`b_f `+ nd`, where $b_f = 16$ – packing basis for overlap factors; `ipd` – pointer to the overlap determinant; `nd` – its power.

29. `C,ij,int,idf` repeat up to the end of file

`C`	angular coefficient for matrix element between angular symmetries `i` and `j`, recorded as `ij=i` × b_c`+j`, where $b_c = 2^{15}$ – packing basis for configurations.
`int`	pointer on the relevant radial integral in the packing form `int = m` × b^8`+k` × b^4`+`i_1 × b^3`+`i_2 × b^2`+`i_3 × `b+`i_4, where `m` – type of integral; `k` – multipole index; i_1, i_2, i_3, i_4 – pointer on the involved orbitals; `b = 10` – packing basis for orbital indexes.
`idf`	pointer on the corresponding overlap factor.

Remarks.

- Only one type of transition can be in the given **mult_bnk**.
- The $\langle\gamma' L'S'\|O^{[\lambda]}\|\gamma LS\rangle$ reduced matrix elements can be obtained from the $\langle\gamma LS\|O^{[\lambda]}\|\gamma' L'S'\rangle$ matrix elements stored in **mult_bnk** by using the factor $(-1)^{L+S-L-'S'}$.
- The present package assumes the $L+S$ coupling order. If one uses atomic wave functions with the $S+L$ coupling order (as in the MCFH_ASP package by Froese Fischer et al. [52], for example), one needs to include the correction factors $(-1)^{L+S-J}$ in both sides.
- The reduced matrix elements in the **mult_bnk** are presented as a linear combination of radial integrals, but not as a summation over weighted one-electron reduced matrix elements, as is done in the MCHF_MLTPOL program [56]. The user can switch **mult_bnk** to the MCHF representation by changing the value of the internal parameter **mltpol** = 0 to **mltpol** = 1.

10. Program BSR_DMAT

The BSR_DMAT program prepares the dipole matrix, **d.nnn**, for subsequent use in the BSR_PHOT program for photoionization calculations, or performs the calculations of oscillator strengths between B-spline bound-state solutions. For completeness, BSR_DMAT also has the option to calculate oscillator strengths between MCHF solutions, and between B-spline bound states and MCHF solutions.

10.1. Dipole transition matrix

For photoionization calculations we need the D-files, which contain the dipole matrix elements between the initial state and the R-matrix solutions for a given partial wave. It is convenient to represent the dipole matrix elements by using the dipole transition matrix, which consists from the dipole matrix elements between the basis states. Suppose we have two sets of solutions of the Hamiltonian matrix in different bases. In our case, it can be either the configuration basis from the MCHF calculations, or a close-coupling B-spline expansion (7.2) from a BSR calculation. Then the matrix element for the given solutions a and b is

$$\langle a|E^{[\lambda]}|b\rangle = \sum_{i,j} a_i d_{ij} b_j = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{d}\boldsymbol{b}, \tag{10.1}$$

where $\{a_i\}$ and $\{b_i\}$ are the corresponding expansion coefficients in the respective bases. Once we generate the dipole transition matrix $\boldsymbol{d}$, the matrix elements for any other solutions is simply defined by convoluting $\boldsymbol{d}$ with the corresponding expansion vectors. The logical structure of BSR_DMAT is such that it first generates the relevant dipole transition matrix, and then, if indicated, calculates the corresponding set of oscillator strengths. This enables us to use the same code for photoionization and bound-state calculations.

Consider the structure of the transition matrix in the B-spline representation. This structure is very similar to the structure of the Hamiltonian matrix considered in the write-up of the BSR_MAT program in Section 7. In the present close-coupling approach, the B-spline bound-state solution vector, $\boldsymbol{c}$, can be written as

$$\boldsymbol{c} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{N_c}, \boldsymbol{b}]^{\mathrm{T}}, \tag{10.2}$$

where each $\boldsymbol{a}_i$ is a column vector of B-spline coefficients for the given channel functions $\bar{\Phi}_i^{\Gamma}$

$$\boldsymbol{a}_i = [a_{1i}, a_{2i}, \ldots, a_{n_s i}]^{\mathrm{T}} \tag{10.3}$$

and $\boldsymbol{b}$ is the column vector of correlation functions coefficients

$$\boldsymbol{b}_j = [b_1, b_2, \ldots, b_{N_p}]^{\mathrm{T}}, \tag{10.4}$$

where N_{c} is the number of channels while N_p is the number of correlation functions. Then the transition matrix $\boldsymbol{d}$ has the form

$$\begin{pmatrix} D(11) & D(12) & \ldots & D(1N_c) & d(1p) \\ D(21) & D(22) & \ldots & D(2N_c) & d(2p) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ D(N_c1) & D(N_c2) & \ldots & D(N_cN_c) & d(N_cp) \\ d(1p)^{\mathrm{T}} & d(2p)^{\mathrm{T}} & \ldots & d(N_cp)^{\mathrm{T}} & d(pp) \end{pmatrix}, \tag{10.5}$$

where $d(pp)$ is an $N_p \times N_p$ matrix that comes from the bound–bound interaction, the $d(ip)$ are the $n_s \times N_p$ matrices, representing the interaction between the ith channel and the bound states, and the $D(ij)$ are the $n_s \times n_s$ matrices for the channel–channel interaction. When we work with MCHF solutions, the transition matrix is reduced to the single last row, if the initial state is given in the MCHF representation, or simply to the $d(pp)$ matrix, if both solutions are MCHF configuration expansions.

Each individual matrix element in (10.5) is expressed by the program MULT in the form

$$\sum c_i R^{\lambda}(a,b) \times D\bigl(\{nl\}; \{n'l'\}\bigr), \tag{10.6}$$

where the c_i are numeric coefficients that only depend on the angular symmetry of the CSFs involved, R^{λ} stands for the corresponding one-electron dipole integrals (9.20), and $D(\{nl\}, \{n'l'\})$ is the overlap factor which depends only on the orthogonality of the radial orbitals used in the construction of the basis CSFs. In general, the overlap factor is the multiplier of determinants of matrices consisting of one-electron overlaps between radial functions in the initial and final states (see Eq. (7.8)).

The B-spline orbitals can appear in the radial integrals R^{λ}, as well as in the determinant factors. To derive the final expression for the matrix elements in the B-spline basis, let us first simplify the overlap factors (7.8) by expanding them over those rows which contain one-electron overlaps with the B-spline orbitals. At most, there can be two such rows. The residual overlap determinants depend only on the known bound orbitals and they can be calculated in a standard manner. It is convenient to redefine the angular coefficients c_i by multiplying them by these residual overlap factors. As a result, the initial overlap factor $D(\{nl\}, \{n'l'\})$ in expression (10.6) is reduced to one-electron or two-electron overlaps of the form

$$\langle kl|nl\rangle, \quad \langle kl|k'l\rangle \quad \text{or} \quad \langle kl|nl\rangle\langle n'l'|k'l'\rangle, \tag{10.7}$$

where $|kl\rangle$ stands for the radial function of the B-spline orbital

$$u_{kl}(r) = \sum_i a_i B_i(r). \tag{10.8}$$

Each term in (10.6) gives rise to some matrix in the spline basis. To elucidate the structure of these matrices, we introduce the following $n_s \times n_s$ matrices with elements

$$B(..)_{ij} = \langle B_i|B_j\rangle, \qquad R(..)_{ij} = \langle B_i|r|B_j\rangle. \tag{10.9}$$

Table 2
Contribution of different terms (without indication of angular coefficients) to the transition matrix. The symbols a, b stand for the bound orbitals, while the symbol k_i indicates the B-spline orbital in channel i

Term	Contribution	Remarks
Channel–channel interaction		
$R(k_i, k_j)$	$R(\cdot\cdot)$	Banded matrix
$\langle k_i\|a\rangle\langle b\|k_j\rangle R^\lambda$	$B(\cdot a)\cdot B(\cdot b) * R^\lambda$ (or L)	Full matrix, contribution due to non-orthogonality
$\langle k_i\|k_j\rangle R^\lambda$	$B(\cdot\cdot) * R^\lambda$	Banded matrix
$R^\lambda(k_i, a)\langle b\|k_j\rangle$	$L(\cdot a)\cdot B(\cdot b)$	Full matrix, contribution due to non-orthogonality
Channel–bound interaction		
$R^\lambda(k_i, a)$	$R^\lambda(\cdot a)$	Vector
$\langle k_i\|a\rangle R^\lambda$	$B(\cdot a) * R^\lambda$	Vector, contribution due to non-orthogonality
Bound–bound interaction		
$R^\lambda(a, b)$		Scalar

To describe the channel–bound interaction, we introduce the following vectors with elements

$$B(.a)_i = \sum_{j=1}^{n_s} a_j B_{ij}, \qquad R(.a)_i = \sum_{j=1}^{n_s} a_j R_{ij}. \tag{10.10}$$

Finally, for bound–bound interaction we have

$$R(ab) = \sum_{i,j=1}^{n_s} a_i R_{ij} b_j. \tag{10.11}$$

With this notation, Table 2 represents the contributions of the different terms from the matrix element expression (10.6), which usually produces the angular integration codes, to the transition matrix (10.5), along with an indication of its structure.

10.2. Oscillator strengths and decay probabilities

For completeness of the description, we provide below the main formulae used in the present code for the calculation of radiative properties. The line strength for a given transition is defined as

$$S_{ij} = S_{ji} = \sum_{m's} \left|\langle J_i M_i | O_\mu^{[\lambda]} | J_j M_j\rangle\right|^2. \tag{10.12}$$

It is given in atomic units, which are $e^2 a_0^{2\lambda}$ and $\beta^2 a_0^{2(\lambda-1)}$ for electric and magnetic transitions, respectively. Here $\beta = e\hbar/2mc$ is the Bohr magneton. The transition probability can be written as:

$$A_{ji} = \frac{2(2\lambda+1)(\lambda+1)}{\lambda[(2\lambda+1)!!]^2} \frac{1}{g_i} \frac{1}{\hbar} \left(\frac{E_j - E_i}{\hbar c}\right)^{2\lambda+1} S_{ij}. \tag{10.13}$$

The oscillator strength (unitless) is related to the transition probability by

$$g_i f_{ij} = \frac{mc}{8\pi^2 e^2} g_j A_{ij} \lambda^2 = \frac{mc^3}{2e^2} \frac{g_j A_{ji}}{\omega^2}, \tag{10.14}$$

independent of the nature of the radiative transition.

10.3. Structure and data flow

The block diagram of the program BSR_DMAT, along with the data flow, is given in Fig. 9. Since BSR_DMAT requires only a small number of input parameters, all of them must be defined in the command line as arguments. The input parameters for the BSR_DMAT program are the names of the c-files with a list of configuration state functions for the initial and final states involved, and the mode of calculation **ctype** (see Section 10.5). The input c-files may be the ordinary c-files from MCHF calculations, or **cfg.nnn** files generated in BSR calculations. The mode of calculation defines whether this is the generation of the transition matrix for subsequent photoionization calculations or the calculation of oscillator strengths between bound states. The **ctype** parameter also defines the representation format for the initial and final states, which can be in the LS- or LSJ-coupling schemes. BSR_DMAT

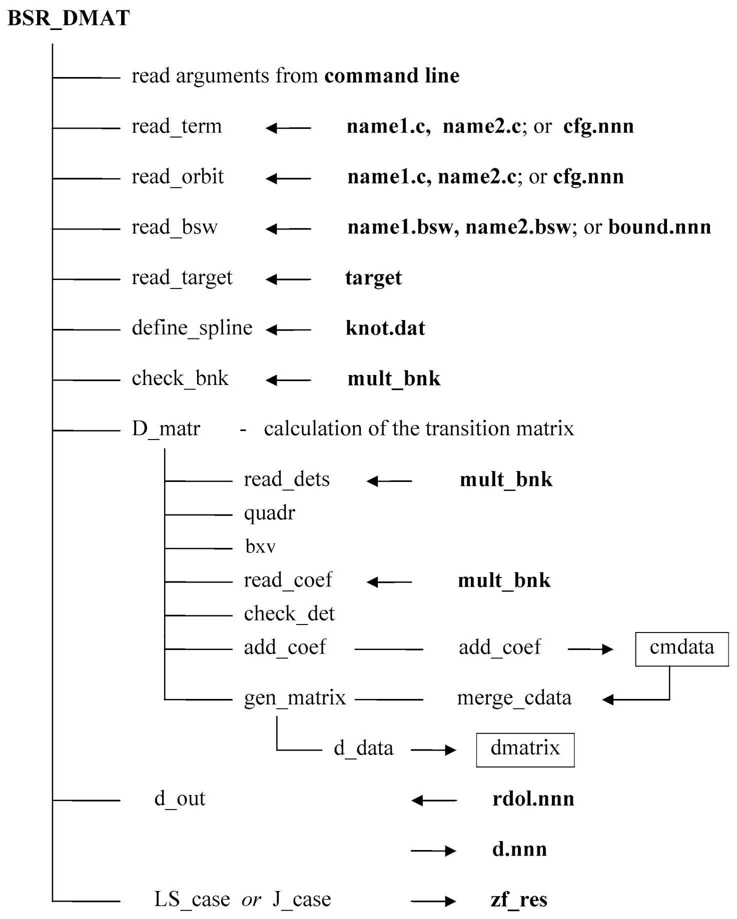

Fig. 9. Block diagram for the program BSR_DMAT and data flow (see text).

reads a set of other input data files. All of them have default names, or names which are directly connected to the names of the input *c*-files.

First, the program defines the set of *LS* terms and one-electron orbitals under consideration (routines **read_term** and **read_orbit**). The orbitals in the initial and final sets are supposed to be non-orthogonal. Then the program reads the one-electron radial functions in the B-spline representation (routine **read_bsw**). They can be either the user-prepared **name1.bsw** files, or the **bound.nnn** files generated by the BSR_HD program. In the latter case, the program also reads the **target** file with a description of the relevant close-coupling expansions. Finally, the program generates all the B-spline arrays needed in the routine **define-spline**. Note that all arrays involved are automatically allocated in accordance with the input data.

In the next step, routine **check_bnk**, the program analyses the information contained in the databank **mult_bnk**, and determines whether the databank contains all angular coefficients needed for the given case. If not, the program stops, suggesting to rerun the MULT program. Note that the multipole index and the type of transition under consideration are defined in **mult_bnk**.

The routine **D_matr** controls the calculation of the transition matrix. First, the routine reads all overlap factors $D(\{nl\},\{n'l'\})$ from the databank in routine **read_dets** before calculating all necessary R^λ integrals in routine **quadr**. All relevant B-spline matrices presented in Table 2 are then generated by the routine **bxv** from the B-spline library. The list of specific integrals (10.6) for the configurations under consideration is generated dynamically from the information in **mult_bnk**. The overlap factor $D(\{nl\},\{n'l'\})$ in expression (10.6) is reduced to one-electron or two-electron overlaps of the form (10.7) by the routine **check_det**. The integrals are then sorted according to their structure (see Table 2) and stored in the module **cmdata** in different ordered blocks (routines **add_coef** and **add_cdata**). When all coefficients from **mult_bnk** are exhausted, or when all blocks are full, they are merged if necessary with the routine **merge_cdata** before being processed by **gen_matrix** to get the B-spline representation for specific integrals (routine **d_data**). The corresponding contributions are then added to the interaction matrix, located in module **dmatrix**.

The next step of the calculation depends on the input mode. In photoionization calculations, the dipole matrix (10.1) is calculated on the base of the R-matrix solutions placed in the file **rsol.nnn**, and then is recorded in the file **d.nnn**. For calculations of oscillator strengths, however, the program calls the routine **LS_case**, or **J_case** for processing transitions between *LSJ* states. The resulting line strengths, oscillator strengths, transition energies, and transition probabilities are recorded in the file **zf_res** (see Section 10.7 below).

10.4. Data files

name1.c *or* **cfg.nnn**	File type: formatted sequential input. Created by MCHF or BSR_CONF programs. Read by routine **check_bnk**. Description: contains the configuration expansion for the initial state.
name2.c *or* **cfg.nnn**	File type: formatted sequential input. Created by MCHF or BSR_CONF programs. Read by routine **check_bnk**. Description: contains the configuration expansion for the final state.
mult_bnk	File type: unformatted sequential input. Written by program MULT. Read by routine **check_bnk**. Description: databank for the angular coefficients of the dipole operator.
bound.nnn	File type: formatted sequential input. Created by program BSR_HD. Read by BSR_DMAT program. Description: bound-state solutions for a given partial wave in the B-spline representation (10.2).
target	File type: formatted sequential input. Written by user and modified by BSR_PREP and BSR_CONF programs. Read by routine **read_target**. Description: contains description of the target states and scattering channels.
knot.dat	File type: formatted sequential input. Written by user. Read by routine **define_grid** from BSPLINE library. Description: input parameters that define the B-spline grid.
target.bsw	File type: unformatted sequential input. Created by program **bsr_prep**. Read by routine **read_bsw**. Description: target one-electron orbitals in the B-spline basis.
pert_nnn.bsw	File type: unformatted sequential input. Created by program **bsr_prep**. Read by routine **read_bsw**. Description: perturber one-electron orbitals in the B-spline basis, optional.
name1.bsw	File type: unformatted sequential input. Created by user. Read by routine **read_bsw**. Description: one-electron orbitals in the B-spline basis for the initial state, optional.
name2.bsw	File type: unformatted sequential input. Created by user. Read by routine **read_bsw**. Description: one-electron orbitals in the B-spline basis for the final state, optional.
d.nnn	File type: formatted sequential output. Written by program BSR_DMAT. Read by BSR_PHOT. Description: dipole matrix.
zf_res	File type: formatted sequential output. Written by program BSR_DMAT. Read by user. Description: oscillator strengths for the dipole operator given in **mult_bnk**.

bsr_dmat.log	File type: formatted sequential output. Written by program BSR_DMAT. Read by user. Description: running information.

10.5. Input parameters

Input parameters must be provided in the command line.

name1.c *or* **cfg.nnn**	name of *c*-file for the initial state.
name2.c *or* **cfg.nnn**	name of *c*-file for the final state.
ctype1	character*1, denotes the representation mode for the initial state: **c** – initial state is defined by *c*-file from MCHF calculations. **l** – initial state is defined by *l*-file from MCHF calculations. **j** – initial state is defined by *j*-file from MCHF calculations. **b** – initial state is defined by **bound.nnn** file from BSR calculations.
ctype2	character*1, the same as **ctype1**, but for the final state. An additional mode is **p**, which indicates that only the dipole transition matrix should be generated in the output file **d.nnn** for subsequent photoionization calculations. All other modes assume the calculation of oscillator strengths, with the output put in the file **zf_res**.

10.6. Output of dipole matrix

Summary of the output records in the **d.nnn** file.

2. `LT1,ST1,IP1,E1` – term and energy of the initial state.
3. `LT2,ST2,IP2,nstate2` – term and number of final states.
4. `(cl(i),cv(i),i=1,nstate2)` – dipole matrix in length and velocity form.

10.7. Output of radiative data in ***zf_res***

The output format in **zf_res** is the same as in the MCHF Atomic Structure Package [52]. An example of output is given in Fig. 10. Each transition is represented by a block, which consists of two empty lines as a delimiter, two lines with a description

```
   -1  -37.77853956  1s(2).2s(2).2p(2)3P2_3P
   -1  -37.48150929  1s(2).2s_2S.2p(3)2D3_3D
    65187.63 CM-1      1534.03 ANGS(VAC)          1534.03 ANGS(AIR)
E1    S =  4.85375D+00   GF = 9.61141D-01        AKI = 1.81637D+08
           6.44181D+00        1.27561D+00              2.41066D+08


   -1  -37.48150929  1s(2).2s_2S.2p(3)2D3_3D
   -1  -37.45433756  1s(2).2s(2).2p_2P.3p_3P
     5963.23 CM-1     16769.43 ANGS(VAC)         16764.85 ANGS(AIR)
E1     S = 6.26544D+00   GF =  1.13495D-01       AKI =  2.99142D+05
           1.17451D+00         2.12756D-02              5.60767D+04


    2 -940.13842975  2p(2)3P2
    2 -932.54958058  2p(4)3P2
  1665544.77 CM-1       60.04 ANGS(VAC)             60.04 ANGS(AIR)
E2     S = 3.55924D-05  GF = 2.76118D-08           AKI = 1.70307D+04


    0 -940.46450303  2p(2)3P2
    2 -940.13842975  2p(2)3P2
    71564.16 CM-1     1397.35 ANGS(VAC)            1397.35 ANGS(AIR)
M1     S = 1.80864D+00  GF =  5.23415D-06          AKI = 5.96024D+03
```

Fig. 10. Example of the output **zf_res** file.

of the initial and final atomic states, one line with the transition energy in cm^{-1} and Angstroms, and finally a line with the line strength, the oscillator strength, and the transition probability in the length form. For E1 transitions, there is an additional line with the results in the velocity form.

Each atomic state is represented by its spectroscopic label, its absolute energy in a.u., and the value of $2J$, in case of a transition between LSJ states. The f-values are always for transitions from the lower to the higher state, whereas the transition probabilities are always reported from the higher to the lower state.

The file **zf_res** is opened in the BSR_DMAT program with access mode *append*. Hence, the results from different calculations can be accumulated.

11. The program BSR_PHOT

The BSR_PHOT program runs the photoionization calculations for one partial wave, based on the **d.nnn** and **h.nnn** files prepared by the programs BSR_DMAT and BSR_HD, respectively. The present implementation of the R-matrix method for photoionization calculations differs from the previous one in that the initial state may be calculated independently from the scattering solutions, with any amount of non-orthogonality between the initial and final one-electron radial functions. We assume that the initial state is well confined to the inner region.

11.1. Structure and data flow

The block diagram of the program BSR_PHOT, along with the data flow, is shown in Fig. 11. All input/output files have prescribed internal names (see Section 11.2). First, the program reads the input parameters from the file **bsr_phot.inp**. The input electron energies are given in Ry and defined in blocks with specific initial energy, energy step, and final energy. The user can define up to ten blocks for a given run that allows to calculate the cross sections with different 'intensity', i.e. some energy regions can be investigated in more detailed, with small energy increments. All output files are opened in the position mode APPEND, and hence all sequential results are added to the output without deleting results from previous calculations. This allows the user to accumulate data from different runs.

The radiative data for a given case are defined in the file **d.nnn**, together with the terms for the initial and final states, as well as the energy of the initial state. Based on the term of the final state, the program looks for the corresponding scattering and R-matrix data in the file **h.nnn**, which may contain data only for one partial wave. Finally, the program reads the internal solution weights

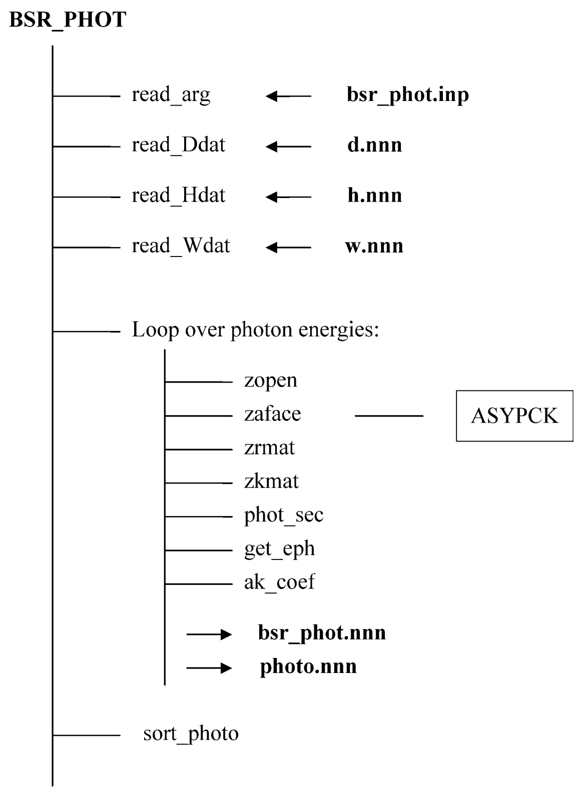

Fig. 11. Block diagram for the program BSR_PHOT and data flow (see text).

from the file **w.nnn**. This optional input is guided by the input parameter **nwt** (see Section 11.3). These data make it possible to define the composition of the final total wavefunction for a given energy, which may be used for detecting and classifying of possible resonance structure.

At the next stage the program executes fully independent calculations of the photoionization cross section for each input energy. The algorithm is given in Section 2.4. In order to determine the photoionization cross section, we need the set of linearly independent solutions (2.33) in the external region. The determination of these solutions is beyond the scope of the present package, which deals with the internal region solutions in the general case. For the treatment of the external region, we currently choose the program ASYPCK [43]. This is a general program, which can determine the solution for both neutral and ionized atoms. It works well over a wide range of energies, including near-threshold regions. ASYPCK is called as a subroutine, which is convenient for its use by another program.

The input data for the external region consist only of the description of the scattering channels and asymptotic coefficients (2.26). Only four main parameters, which are given in Section 11.3, guide the execution of ASYPCK. For a more detailed description of the ASYPCK package, the user is referred to its write-up and that of the program IMPACT [36]. The ASYPCK program is called in the present calculation by using the interface subroutine **zaface** (see Fig. 11). We did not introduce any modifications to ASYPCK itself, and hence the user should check if the dimension parameters in ASYPCK are big enough for a given run. Over the past years, a set of more effective algorithms were developed for the treatment of the external region, such as the flexible asymptotic R-matrix package FARM [28] or a new version of the program STGF [59]. In principle, ASYPCK can easily be replaced by any other algorithm, provided the latter is organized as a subroutine call.

The photoionization calculations for a given energy consist of the following steps. The routine **zopen** defines the number of open channels and the corresponding channels energies. Then routine **zaface** calls the program ASYPCK to determine the asymptotic radial functions (2.27) at the R-matrix radius. The routines **zrmat**, **zkmat** and **get_eph** calculate the R- and K-matrices, together with the eigenphase sum at the energy of interest. Then the total dipole matrix elements (2.31), and the partial and total cross sections are calculated by the routine **phot.sec**. Optionally, the routine **ak_coef** is called if one needs the channel composition for the total wave function.

The results are saved in a set of output files. The file **photo.nnn** contains the total cross section and the accuracy of the calculations, defined via the average asymmetry of the K-matrix. The results in this file are sorted according to energy by the routine **sort_photo**. The file **photo.nnn** is in simple table form and can be used for a quick check of the calculations. The full results including partial cross sections, eigenphase sums, channel weights and full dipole matrix elements (2.31) are recorded in the file **bsr_phot.nnn**. These data may then be used for the calculation of differential cross sections or the asymmetry parameter β (see the description of utility-program **photo_tab** in Section 13).

11.2. Data files

bsr_phot.inp	File type: formatted sequential input. Created by the user. Read by the routine **read_arg**. Description: input parameter for given run.
d.nnn	File type: unformatted sequential input. Created by BSR_DMAT program. Read by the routine **read_Ddat**. Description: radiative dipole transition matrix for given initial state and final scattering partial wave.
h.nnn	File type: unformatted sequential input. Created by the BSR_HD program. Read by routine **read_Hdat**. Description: diagonalized Hamiltonian matrix data for the inner region.
w.nnn	File type: unformatted sequential input. Written by the program BSR_HD. Read by the routine **read_Wdat**. Description: weights of different channels in the inner-region solutions.
photo.nnn	File type: formatted sequential output. Created by the program BSR_PHOT. Read by the user. Description: electron and photon energies, total cross-section in the length and velocity forms, eigenphase and accuracy of the calculations, defined as the average asymmetry of the K-matrix.

bsr_phot.nnn	File type: formatted sequential output. Created by the program BSR_PHOT. Description: partial cross sections, eigenphases, dipole matrix elements (2.31) and channel weights for each input energy.
bsr_phot.log	File type: formatted sequential output. Created by the program BSR_PHOT. Read by the user. Description: running information.

11.3. *Input parameters*

The input parameters are provided in the file **bsr_phot.inp**. All input data have the format '*name* = *value*', and should be placed at the beginning of the line; otherwise they will be ignored. The default values of the data are indicated in the brackets. All default values, along with the unit numbers and the default file names are placed in the module **mod_phot**. Some of the input parameters deal with the asymptotic package ASYPCK.

klsp	index of partial wave (**nnn**).
elow **estep** **ehigh**	Define the range of input electron energies (in Ry) as the initial energy (**elow**), the energy increment (**estep**), and the final energy (**ehigh**). This can be given for up to ten different input energy intervals.
awt [100]	atomic number, used to convert Ry to eV.
ibug [0]	defines the level of debug printing in the file **bsr_phot.out**.
e_exp [0.0]	if $\neq 0$, define the ionization (detachment) energy.
nwt [0]	if > 0, then the weights of each channel in the total solutions are also calculated, based on the data in the file **w.dat**.

Parameters for ASYPCK program:

AC [0.001]	accuracy for the asymptotic solutions.
DR [0.1]	in ASYPCK, the solutions are first obtained at R1=R-DR and R2=R+DR, and then by interpolation – in R.
mfg [300]	an asymptotic expansion of the solution will be sought at R + DR*MFG.
iauto [2]	if > 0, then ASYPCK may automatically increase parameter MFG.

12. Libraries

The BSR package includes the four libraries BSPLINE, ZCONF, ZCOM and LAPACK, which contain routines common to all programs. The ZCONF library contains specific routines, which deal with the configuration and close-coupling expansions, while ZCOM contains common routines used for a variety of auxiliary computations. The LAPACK library (http://www.netlib.org/lapack/index.html) was chosen to provide the necessary linear algebra calculations, such as diagonalizing matrices or solving a system of linear equations.

The BSPLINE library is the central part of the present complex. It contains routines for using B-spline bases in atomic structure calculations. Extensive survey of recent applications of B-splines in atomic and molecular physics is presented in the review by Bachau et al. [26]. The present library represents the accumulated developments made by different people over the last twenty years. BSPLINE includes original FORTRAN IV routines from the monograph by de Boor [60], who first presented algorithms for the construction and manipulation of B-splines. These were modified by Froese Fischer and her collaborators who started the design and implementation of the present library, and numerous investigations of different numerical algorithms for using B-splines in atomic structure calculations were published [39,49,50,61–64]. This spline library has been modified and extended by the present author. Features were added to accommodate the more complex requirements of the R-matrix method, and the programming was upgraded to F95 standard.

Most of the routines in BSPLINE can be used in many other atomic structure calculations. For this reason we decided to present a rather detailed description of the BSPLINE library, including calling sequences for the main routines and the internal structure of the modules. The FORTRAN module is a program unit, which contains specifications and definitions that can be made accessible to other program units. It was found convenient to cluster the principal parameters, variables and arrays in different modules according

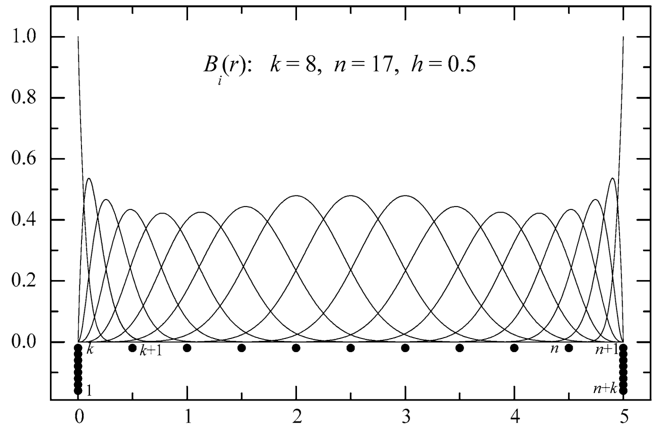

Fig. 12. The full set of B-splines of order $k = 8$ relative to the equidistant knot sequence (represented by full circles, with an indication of their numeration).

to the subject they describe. For the given module *name* to be accessible, the other program units must reference the module in a *use name* statement. All arrays in the modules in the present implementation are dynamically allocated for the case under consideration. The module structure used in BSPLINE simplifies the use of separate subroutines in other applications.

The modules and routines in the description below have been grouped together for convenience. This includes the general routines and the atomic-orbital-related routines. The latter concern first of all the evaluation of a two-electron integral in the B-spline basis, and the generation of the B-spline representation for the Hamiltonian matrix. Along with the description of the routines, each section provides a brief description of the basic theory.

12.1. Description of B-splines

B-splines are functions designed to generalize polynomials used for approximating arbitrary functions in some finite interval. Consider the interval $[a, b]$ divided into subintervals. The endpoints of these subintervals are given by the knot sequence $[t_i]$, $i = 1, 2, \dots, n + k$. The B-splines of order k, $B_{i,k}(r)$, on this knot sequence are defined recursively by the relations

$$B_{i,1} = \begin{cases} 1, & t_i \leqslant r \leqslant t_{i+1}, \\ 0, & \text{otherwise} \end{cases} \tag{12.1}$$

and

$$B_{i,k}(r) = \frac{r - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(r) + \frac{t_{i+k} - r}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(r). \tag{12.2}$$

Since k and t are usually fixed, we will indicate only one index, B_i. Each B_i is defined over the interval $[t_i, t_{i+k}]$, which contains $k + 1$ consecutive knots, and is indexed by the knot where it starts. An example of a full set of B-splines is shown in Fig. 12. In this example, the interval $[0, 5]$ is divided into equidistant subintervals, with $h = 0.5$. Each B_i starts at the knot t_i, $i = 1, \dots, n$, and ends k knots later. In order to provide the same number of B-splines in each interval, the multiplicity of knots at the endpoints is usually chosen as the maximum possible value, which is equal to the B-spline order k. The most common choice for the multiplicity at the inner knots is unity, corresponding to the maximum continuity of B-spline functions inside the interval. With this choice, employed in the BSPLINE library, the number of subintervals is related to the number of B-splines as

$$n_{\text{int}} = n + 1 - k.$$

Some general properties of B-splines are the following:

- Over each subinterval $[t_i, t_i + 1]$, exactly kB-splines are nonzero

  $$B_j(x) \neq 0 \quad \text{for } j = i - k + 1, \dots, i,$$

  the first being B_{i-k+1}, which ends at $t_i + 1$, and the last being B_i, which starts at t_i. This means that

  $$B_i(x) \cdot B_j(x) = 0 \quad \text{for } |i - j| \geqslant k.$$

- In the expansion of an arbitrary function

  $$f(x) = \sum_{j=1}^{n} c_j B_j(x) = \sum_{j=i-k+1}^{i} c_j B_j(x) \quad \text{for } x \in [t_i, t_{i+1}]$$

only k terms contribute. Consequently, the B-spline representation for different operators will exhibit a band structure, simplifying to a great extent the solution of the relevant matrix equations.

- B-splines are normalized as $\sum_i B_i(x) = 1$. Since the B-splines are non-negative with minimal support, the expansion coefficients of an arbitrary function are close to the function values at the knots. This means that wild oscillations in the coefficients are avoided, cancellation errors are minimal, and the numerical stability is maximized.
- B-splines satisfy the recursion relations (12.2). Together with the definition of B-splines of order $k = 1$ (12.1), this provides a simple algorithm for their practical evaluation: given a point x, one generates by recursion the values of all kB-splines that are nonzero at x. The derivative of a B-spline of order k can also be expressed as a linear combination of B-splines of order $(k-1)$:

$$DB_{i,k}(r) = \frac{k-1}{t_{i+k-1} - t_i} B_{i,k-1}(r) - \frac{k-1}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(r). \tag{12.3}$$

  The first B_1-spline is the only spline with nonzero value at the left end of interval $[a,b]$ while all other B_i-splines have $(r-a)^{i-1}$ dependence. A similar behavior applies at the point b. This makes it very easy to implement boundary conditions at the endpoints. For instance, $f(a) = 0$ is satisfied by deleting B_1 from the set, or $Df(b) = 0$ may be satisfied by combining B_{n-1} and B_n as $\tilde{B}_{n-1} = c_1 B_{n-1} + c_2 B_n$ so that $D\tilde{B}_{n-1}(b) = 0$, and so on.
- With a minimum multiplicity of knots, the $B_{i,k}$-spline functions belong to the functions of class C^{k-2}, i.e. they are all continuous functions together with their derivatives up to order $k-2$. In each subinterval $[t_i, t_{i+k}]$, B-spline functions can be represented by a polynomial of order k (maximum degree $k-1$). That allows, in principle, for analytical manipulations with B-spline functions in symbolic packages.

In quantum mechanical applications we deal mostly with matrix elements, i.e. definite integrals involving B-splines and their derivatives. These are best computed to machine accuracy by employing Gauss–Legendre integration of appropriate order over each subinterval $[t_i, t_{i+1}]$. Recall that the Gauss–Legendre method of order n (n points) is exact for a polynomial integrand of order $2n$. This means that a k-points Gauss–Legendre method integrates exactly the product $B_i B_j$, and also provides accurate results even for non-polynomial integrands, provided no singularity is present in the given interval. At the same time, in the first subinterval, $(1/r)B_i B_j$ is also evaluated exactly if i or $j > 1$. For most applications, therefore, we only need the B-spline values at the k Gauss points. This also spares us from directly dealing with the vicinity of $r = 0$, where the potentials may have singularities.

Another important problem in quantum mechanical applications is the solution of the radial Schrödinger equation. In order to demonstrate the use of B-splines for this problem, consider the one-electron radial equation

$$\left[-\frac{\mathrm{d}^2}{2\mathrm{d}r^2} + \frac{l(l+1)}{2r^2} + V(r)\right] u_{nl}(r) = E_{nl} u_{nl}(r). \tag{12.4}$$

An approximation solution is expressed as a linear combination of B-splines over an interval $[0, r_{\max}]$, say $\{B_i(r), i = 1, \dots, n_s\}$. Then

$$u_{nl}(r) = \sum_{i=1}^{n_s} c_i^{nl} B_i(r). \tag{12.5}$$

Usually the approximate (or trial) functions satisfy the boundary conditions. When the B-spline basis is generated using a grid with multiple knots at the ends of the interval, the zero boundary conditions, for example, are fulfilled by removing the first and the last B-splines from the basis set. We shall refer to the column vector $\boldsymbol{c} = \{c_i\}$ as a spline expansion vector of $u(r)$. The solutions of (12.4) are calculated by solving the system of n_s linear equations obtained when substituting (12.5) into (12.4) and projecting onto $B_j(r)$. Written in matrix form, this procedure is equivalent to solving the generalized eigensystem

$$\boldsymbol{H} \cdot \boldsymbol{c} = E\boldsymbol{S} \cdot \boldsymbol{c}, \tag{12.6}$$

where

$$H_{ij} = -\frac{1}{2}\langle B_i|\frac{\mathrm{d}^2}{\mathrm{d}r^2}|B_j\rangle + \frac{l(l+1)}{2}\langle B_i|\frac{1}{r^2}|B_j\rangle + \langle B_i|V|B_j\rangle, \tag{12.7}$$

$$S_{ij} = \langle B_i|B_j\rangle. \tag{12.8}$$

The overlap matrix $\boldsymbol{S}$ originates from the fact that the B-splines do not form an orthogonal set of basis functions. We will refer to the H_{ij} matrix as the B-spline representation of the Hamiltonian. Note that all matrix elements in (12.6) may be easily computed, as discussed above, to machine accuracy using Gauss–Legendre integration, applied to each subinterval. Another key property related to the use of a B-spline basis follows from the fact that a B-spline of order k differs from zero only on k successive subintervals. Therefore, any local operator expressed in the B-spline basis appears as a banded matrix with nonvanishing values only on a diagonal band of width $2k-1$. This simplifies considerably the solution of the generalized eigenproblem (12.6).

Provided the wavefunctions fit into the cavity (often called the box), in which we solve the problem (12.6), the first few eigenvalues and eigenvectors agree precisely with the corresponding bound-state eigenvalues and eigenvectors obtained by numerically integrating the radial Schrödinger equation, but as the principal quantum number increases, the cavity spectrum departs from the real spectrum and forms the so-called *pseudospectrum*, which can be used instead of the real spectrum in different physical problems. Each of these eigenvectors defines an atomic orbital. By using a B-spline set, it is possible to account for the influence of an entire Rydberg series without introducing a cut-off. For example, the second-order energy correction in the framework of the Rayleigh–Schrödinger perturbation theory

$$\Delta E^{(2)} = \sum_{n\neq 0} \frac{\langle 0|V|n\rangle\langle n|V|0\rangle}{E_n - E_0} \tag{12.9}$$

requires the direct evaluation of the sum in terms of an infinite sum over the bound states and an integration over the continuum. In the B-spline pseudospectrum, only a few states are bound while the remaining states are unbound, and the infinite sum (12.9) is reduced to a finite sum. As shown in many numerical examples [65], this sum very accurately reproduces the infinite sum (12.9) with a small number of B-splines, i.e. the B-spline basis is *effectively complete* in this sense. This effective completeness of the basis set means that the summation is independent, or almost independent, of the box size and the number of splines if these two quantities are chosen sufficiently large. We can check numerically if the summation is independent of the box size and the number of splines. Completeness of the B-spline basis is a very valuable property, simply because it is possible to replace an infinite summation over bound states and an integration over continuum states with a summation over a small number of L^2-integrable functions. This property also found wide application in the investigation of the strong interaction between a perturber and a Rydberg series [66], or in the investigation of the influence of the continuum spectrum in two-electron atoms [67].

12.2. General routines

B-splines can be considered as elementary functions like $\sin(x)$ or $j_l(x)$. One becomes familiar with them by understanding their qualitative behavior and how to use them, i.e. how to obtain values of the functions, their derivatives, or integrals. This is the purpose of the routines described in the next section.

List of general routines

bav	computes $\mathbf{y} = \mathbf{Bx}$ or $\mathbf{y} = \mathbf{xB}$, where $\mathbf{B}$ is symmetric or non-symmetric band matrix, $\mathbf{y}$ and $\mathbf{x}$ are vectors.
bvalue	calculates at given radius the value and derivatives of a function given in a B-spline representation.
bvmv	returns the matrix element $\langle \mathbf{x}|\mathbf{A}|\mathbf{y}\rangle$, where $\mathbf{x}$, $\mathbf{y}$ are vectors, $\mathbf{A}$ is a band array in symmetric, non-symmetric, or almost symmetric form.
bxv	computes $\mathbf{y} = \mathbf{Bx}$, where $\mathbf{B}$ is a symmetric band matrix while $\mathbf{y}$ and $\mathbf{x}$ are vectors.
define_grid	gets input data for the grid and sets up the knots for spline.
define_spline	initializes the values of the spline and its derivatives, and evaluates the basic elementary operators in the spline basis.
dinty	computes the matrix elements $\langle B_i|y(r)|B'_j\rangle$ between individual B-splines for an input function $y(r)$ given at the Gaussian points of each interval.
facsb	factorizes the overlap matrix $\langle B_i|B_j\rangle$.
gauss	provides the Gaussian coordinates and weights over the interval [0, 1].
minty	computes the matrix elements $\langle B_i|y(r)|B_j\rangle$ between individual B-splines for an input function $y(r)$ given at the Gaussian points of each interval.
mkgrid	sets up the knot sequence.
mrm	generates the B-spline representation of the operator r^m.
spline_param	module, contains the main spline parameters.
spline_grid	module, defines the values of the splines and their most general functions at the Gaussian points needed for integrations on the grid.

spline_galerkin	module, contains common arrays used in the application of splines in the Galerkin method.
splin3	provides coefficients for a cubic interpolating spline.
vbsplvd	calculates the values of the B-splines and their derivatives for one point per each interval.
vinty	computes the vector elements $\langle B_i \vert y(r)\rangle$, where the function $y(r)$ is given at the Gaussian points of each interval.
yval	computes the values of $r^m f(r)$, $r^m f'(r)$ or $r^m f''(r)$ at the Gaussian points of each interval, where $f(r)$ is defined by the spline expansion vector.

Internal representation of band matrices

The BSPLINE library uses different storage modes for banded matrices depending on the symmetry of the matrix and on the operator represented by this matrix. Let $A(n,n)$ be a matrix with a diagonal band of width $2k-1$. Then the possible storage modes, matrices B, can be described schematically as:

- **Symmetric lower-column storage mode**

$$\boldsymbol{A}(n,n) \to A_{ij} = \begin{vmatrix} 1 & & & & \\ \bullet & 1 & & & \\ k & \bullet & 1 & & \\ & k & \bullet & 1 & \\ & & k & \bullet & 1 \end{vmatrix}, \qquad \boldsymbol{B}(n,k) \to B_{i,j'} = \begin{vmatrix} & & 1 \\ & \bullet & 1 \\ k & \bullet & 1 \\ k & \bullet & 1 \\ k & \bullet & 1 \end{vmatrix},$$

$$\boldsymbol{A}(i,j) = \boldsymbol{B}(i,j') \quad \text{with } j = j' + i - k.$$

The summation over elements:
```
Do j'= 1,k
Do i = k+1-j',n
```

- **Symmetric upper-column storage mode**

$$\boldsymbol{A}(n,n) \to A_{ij} = \begin{vmatrix} 1 & \bullet & k & & \\ & 1 & \bullet & k & \\ & & 1 & \bullet & k \\ & & & 1 & \bullet \\ & & & & 1 \end{vmatrix}, \qquad \boldsymbol{B}(n,k) \to B_{i,j'} = \begin{vmatrix} 1 & \bullet & k \\ 1 & \bullet & k \\ 1 & \bullet & k \\ 1 & \bullet & \\ 1 & & \end{vmatrix},$$

$$\boldsymbol{A}(i,j) = \boldsymbol{B}(i,j') \quad \text{with } j = j' + i - 1.$$

The summation over elements:
```
Do j'= 1,k
Do i = 1,n-j'+1
```

- **Non-symmetric column storage mode**

$$\boldsymbol{A}(n,n) \to A_{ij} = \begin{vmatrix} 1 & \bullet & k & & \\ \bullet & 1 & \bullet & k & \\ k & \bullet & 1 & \bullet & k \\ & k & \bullet & 1 & \bullet \\ & & k & \bullet & 1 \end{vmatrix}, \qquad \boldsymbol{B}(n,2k-1) \to B_{i,j'} = \begin{vmatrix} & & 1 & \bullet & k \\ & \bullet & 1 & \bullet & k \\ k & \bullet & 1 & \bullet & k \\ k & \bullet & 1 & \bullet & \\ k & \bullet & 1 & & \end{vmatrix},$$

$$\boldsymbol{A}(i,j) = \boldsymbol{B}(i,j') \quad \text{with } j = j' + i - k.$$

The summation over elements:
```
Do j'= 1,k+k-1
Do i = max(1,1+k-j'),min(n,n+k-j')
```

- **Almost symmetric lower-column storage mode**
  There is only one non-symmetric element: $A(n,n-1) \neq A(n-1,n)$, the latter is saved in $B(1,1)$ for low-column storage mode, or in $B(n,n)$ for upper-column storage mode.

MODULE spline_param

ks	order of B-splines.
ns	number of B-splines.

nv	number of intervals.
ml	number of distinct points from 0 to 1.
me	number of points in the "exponential" region.
h	initial step in knot sequence.
hmax	maximum step.
rmax	border radius.

MODULE spline_grid

t`(1:ns+ks)`	knot sequence.
bsp`(1:nv+1,1:ks,1:ks)`	`bsp(i,m,ith)` contains value of the `i+ith-1` B-spline in interval i at Gaussian point m.
bspd`(1:nv+1,1:ks,1:ks,1|2)`	values of first and second derivatives of B-splines at the same points. `bsp(nv+1,1,.)` and `bspd(nv+1,1,.,.)` contain the corresponding values at the last knot point, i.e. at $r_{\max}$.
bsq`(1:nv+1,1:ks,1:ks)`	values of the expression $B(r) - 1/r B'(r)$ at the Gaussian points.
gr`(1:nv,1:ks)`	`gr(i,m)` contains Gaussian point m in interval i.
grm`(1:nv,1:ks)`	reciprocals of the values of `gr(i,m)`.
grw`(1:nv,1:ks)`	Gaussian weights at the corresponding points.

Internal routines:

Allocate_grid	allocates space for arrays `bsp`, `bspd`, `gr`, `grm`, and `grw` according to parameters from module **spline_param**.

MODULE spline_galerkin

sb`(1:ns,1:ks)`	`sb(i,j) -> <`B_i`|`B_{i+j-k}`>`
r1`(1:ns,1:ks)`	`r1(i,j) -> <`B_i`| r|`B_{i+j-k}`>`
rm1`(1:ns,1:ks)`	`rm1(i,j) -> <`B_i`|(1/r)|`B_{i+j-k}`>`
rm2`(1:ns,1:ks)`	`rm2(i,j) -> <`B_i`|(1/`r^2`)|`B_{i+j-k}`>`
rm3`(1:ns,1:ks)`	`rm2(i,j) -> <`B_i`|(1/`r^3`)|`B_{i+j-k}`>`
db1`(1:ns,1:ks)`	`db1(i,j) -> <`B_i`|`B'_{i+j-k}`>`
db2`(1:ns,1:ks)`	`db2(i,j) -> <`B_i`|`B''_{i+j-k}`>`
bs`(1:ks,1:ns)`	factorization of array `sb`

Internal routines:

Allocate_galerkin	allocates space for all arrays according to the parameters from module **spline_param**. The arrays are stored in the symmetric lower-column storage mode, except for `db2` – almost symmetric mode, and `db1` – antisymmetric array.

Remarks. All arrays and parameters in the above modules should be initialized before the B-spline applications. This can be done with the routines **define_grid** and **define_spline**, which should be called at the beginning of each BSR program.

The **define_grid** subroutine generates the knot sequence. This is a very important first step in the BSR calculations, because the B-splines are fully defined by the given knot sequence. Hence the grid, together with the spline order, k, determines the accuracy of the approximation. Though there is complete freedom in choosing the mesh of knots, the optimal choice will depend on the type of result we are particularly interested in. With splines, we can create any composite grid if needed but too dense a grid can lead to a rapid saturation of the computational resources. Experience shows that the most appropriate choice is a logarithmic grid that reflects the exponential behavior of atomic orbitals. On the other hand, in the continuum calculations, the wavelength of the scattering particle cannot be smaller than the grid step; otherwise the B-spline basis hardly describes the oscillating behavior of the wavefunction. In the present implementation, we use a mixed sequence which can easily be adjusted to the task under consideration.

We introduce four main parameters: the step size at the origin, $h = 2^{-m}$, where m is the integer, the maximum step size $h_{\max}$, the maximum range $r_{\max}$, and the order of splines k. Let Z be the nuclear charge of the atom. The grid points are now defined through the array $t_i = Zr_i$ such that

$$\begin{aligned}
&t_i = 0 && \text{for } i = 1, \ldots, k,\\
&t_{i+1} = t_i + h && \text{for } i = k, \ldots, k+m,\\
&t_{i+1} = t_i(1+h) && \text{for } 1 < t_{i+1} - t_i < h_{\max},\\
&t_{i+1} = t_i + h_{\max} && \text{for the rest of } t_i,\\
&t_i = r_{\max}/\mathrm{Z} && \text{for } i = n, \ldots, n+k.
\end{aligned} \tag{12.10}$$

This grid is linear near the origin, at $Zr < 1$, which guarantees an accurate representation of the radial orbitals at small radii. Experience shows that values $\mathrm{Z}h = 0.25$ or 0.125 are good enough for most applications. In the middle-radii region, the grid is exponential. This gives some advantage in computational time, because for B-splines entirely in the outer region, relations such as

$$\langle B_i, r^p B_i\rangle = (1+h)^{p+1}\langle B_{i-1}, r^p B_{i-1}\rangle \tag{12.11}$$

hold (see, for example, [39]). Similar rules can also be derived for two-electron integrals commonly occurring in atomic structure calculations (see Section 12.4). Thus, only a few integrals need to be evaluated while a scaling law applies for others.

Another grid parameter defines the maximum range of the solution. Since the range of a 1s orbital is much less than that of a 10s orbital, the software may automatically adjust the range for each orbital, but in a general purpose software, a sufficient range is needed to allow orbitals to reach their full extent for the desired accuracy. In BSR calculations, $r_{\max}$ is usually chosen from the condition that $|P(r_{\max})| < 10^{-5}$ for all bound radial functions, and this value coincides with the R-matrix boundary, a.

For bound-state calculations we can use the exponential grid in the outer region, whereas in scattering calculations the maximum size of the subintervals should be restricted according to the maximum energy of the scattering electron. Experience shows that the oscillating behavior of the scattering radial functions is reproduced accurately with at least three knot points for a half wavelength. Hence, the user should choose the parameter $h_{\max} \leqslant 1/k_{\max}$, where $k_{\max}$ is the maximum linear momentum of the scattering electron expected in the subsequent calculations. This restriction is primarily important in direct scattering calculations [42], whereas

```
               8   ==>   order of splines (ks)
             101   ==>   number of splines (ns)
         6.00000   ==>   nuclear charge (z)
         0.25000   ==>   step size from 0 to 1 (h for z*r, = 1/2\m)
         0.70000   ==>   maximum step size (hmax for r)
        50.00000   ==>   maximum r (rmax)
***
         0.00000         0.00000    0.00000    0.00000    0.00000
         0.00000         0.00000    0.00000    0.04167    0.08333
         0.12500         0.16667    0.20833    0.26042    0.32552
         0.40690         0.50863    0.63578    0.79473    0.99341
         1.24176         1.55220    1.94026    2.42532    3.03165
         3.63798         4.24431    4.85064    5.45697    6.06330
         6.66963         7.27596    7.88229    8.48862    9.09495
         9.70128        10.30761   10.91394   11.52027   12.12660
        12.73293        13.33926   13.94559   14.55192   15.15825
        15.76457        16.37090   16.97723   17.58356   18.18989
        18.79622        19.40255   20.00888   20.61521   21.22154
        21.82787        22.43420   23.04053   23.64686   24.25319
        24.85952        25.46585   26.07218   26.67851   27.28484
        27.89117        28.49750   29.10383   29.71016   30.31649
        30.92282        31.52915   32.13548   32.74181   33.34814
        33.95447        34.56080   35.16713   35.77346   36.37979
        36.98612        37.59245   38.19878   38.80511   39.41144
        40.01777        40.62410   41.23043   41.83676   42.44309
        43.04942        43.65575   44.26208   44.86841   45.47474
        46.08106        46.68739   47.29372   47.90005   48.50638
        49.11271        50.00000   50.00000   50.00000   50.00000
        50.00000        50.00000   50.00000   50.00000
***
```

Fig. 13. Example of a **knot.dat** file in the case of the e–C scattering.

in the R-matrix calculations was found that this restriction can be reduced at least three times without noticeable influence on the numerical accuracy.

The grid parameters are provided in the BSR calculations in the file **knot.dat**, and these parameters should hold the same values during the entire set of calculations. An example of a **knot.dat** file is given in Fig. 13. The routine **define_grid** first reads the main parameters from **knot.dat**, and then generates the knot sequence in the routine **mkgrid**. The number of B-splines is an output parameter in this procedure and defined at this stage. The resulting knot sequence and the relevant parameters are then rewritten to the **knot.dat** file. If the program cannot find the **knot.dat** file, the execution is halted. The user is strongly advised to first try different sets of parameters to produce the desired knot sequence with a minimum number of B-splines, because this parameter strongly affects the computation time for all subsequent calculations.

12.3. Atomic orbital routines

This block of routines deals with the set of atomic radial functions and their B-spline representations, denoted below as vectors $\mathbf{p}_i$. The operator $\mathbf{L}$ is defined as

$$L = \frac{d^2}{dr^2} - \frac{l(l+1)}{r^2} + \frac{z}{r}.$$

List of routines

azl	provides B_{l+2}/r^{l+1} at $r = 0$, which defines the leading term for orbital nl.
bcore	computes the energy of the common closed shells (core).
bhl	evaluates the matrix element $\langle \mathbf{p}_i \vert L \vert \mathbf{p}_j \rangle$ for atomic orbitals i, j.
bhwf	computes the spline expansion coefficients for hydrogen radial functions.
coulomb	builds the interaction matrix for the Coulomb problem.
grad	gives the matrix element $\langle \mathbf{p}_i \vert d/dr + l/r \vert \mathbf{p}_j >$ for atomic orbitals i, j.
hlc	computes the matrix element $\langle \mathbf{p}_i \vert L \vert \mathbf{p}_j \rangle$ modified by the interaction with closed shells.
hlm	sets up the matrix of the **L** operator in the B-spline basis.
quadr	evaluates the matrix element $\langle \mathbf{p}_i \vert r_k \vert \mathbf{p}_j \rangle$ for atomic orbitals i, j.
mvcv	computes the matrix elements for the mass-velocity correction in the B-spline basis.
r_bwfn	reads the atomic orbitals from the *bsw*-files.
spline_atomic	module, contains the main atomic parameters.
spline_hl	module, contains the **L** matrix and routines for evaluation of relativistic shifts.
spline_orbital	module, contains the description of the atomic orbitals.
zeta	evaluates the spin–orbit integrals (including the interaction with closed shells).

MODULE spline_orbitals

nbf	number of atomic orbitals.
nbs`(1:nbf)`	principal quantum numbers.
lbs`(1:nbf)`	orbital angular momenta of the orbitals.
kbs`(1:nbf)`	set indexes of orbitals.
ebs`(1:nbf)`	spectroscopic labels for orbitals.
mbs`(1:nbf)`	length of the B-spline expansion.
iech`(1:nbf)`	Additional pointer, used to distinguish the continuum orbitals.
pbs`(1:ns,1:nbf)`	B-spline expansion coefficients for atomic orbitals.

qbs`(1:ns,1:nbf)`	vectors $\boldsymbol{B} \cdot \boldsymbol{p}$, where $\boldsymbol{B}$ is the B-spline overlap matrix.
obs`(1:nbf,1:nbf)`	one-electron overlaps.
iort`(1:nbf,1:nbf)`	orthogonal constraints, imposed on the one-electron orbitals.

Internal routines:

allocate_bsorb	allocates space for all arrays for a given value of `nbf`.
Ifind_bsorb	defines the position of the orbital n, l, k in the common list.
Iadd_bsorb	adds the orbital n, l, k to the common list.

MODULE spline_atomic

z	charge of nucleus.
kclosd	number of closed shells.
rel	indicates a relativistic calculation.
fine	fine-structure constant.
EC	energy of core.
ioo	if ioo > 0, the core energy also includes the orbit–orbit interaction.

MODULE spline_hl

hl`(1:ns,1:ks)`	matrix of the **L** operator in the B-spline basis (in almost symmetric lower-column mode).
lh	the corresponding value of the orbital angular momentum.
vc`(1:ns,1:ks)`	matrix of mass-velocity correction in the B-spline basis.

Internal routines:

Allocate_hl	allocates space of for the hl and vc arrays according to parameters from module *spline_param*.

Format of the *bsw*-files. The atomic orbitals in the B-spline representation are supposed to be recorded in the unformatted *bsw*-files, which are similar to the w-files in the MCHF package [68]. In the *bsw*-files each orbital is recorded sequentially with the following information:

```
1. ebs, z, h, hmax, rmax, ksb, nsb, mbs
2. pbs(1:mbs)
```

where the notation corresponds to the definitions in the module spline_orbitals. Along with the usual information about the quantum numbers and the B-spline expansions, the *bsw*-file also contains the B-spline parameters, which enables us to reconstruct the corresponding knot sequence.

12.4. Evaluation of two-electron integrals

The BSPLINE library contains three sets of routines for the evaluation of two-electron integrals, which are based on the different numerical methods. They include the usual direct differential method, the B-spline differential method, and the direct summation over the B-spline cells. The different methods are suitable for different kind of calculations, have different accuracy and speed, and different representations of the final results. The direct differential method provides only the total values for the integrals, has the lowest accuracy but the highest speed. This method can be used in large-scale CI calculations. The B-spline differential method provides the B-spline representation for different two-electron integrals, that can be used in generating the Hamiltonian matrix, including the Breit–Pauli terms. The evaluation of the total values of integrals, however, is slower, approximately by one order of magnitude, than in the direct differential method. In the direct summation over B-spline cells, we first evaluate the two-electron integrals in the different (r_1, r_2) cells, defined by the B-splines subintervals $[t_i, t_{i+1}]$. The integration is performed by using the Gauss–Legendre quadratures. This provides the highest accuracy, which is comparable to machine accuracy. This method can be used efficiently for both the B-spline representation and the evaluation of total integral values, although the speed is still slower that in the direct differential method.

Below we outline the relevant theory and the different numerical methods used in BSPLINE to deal with the two-electron integrals. Then we give a short description of the relevant routines and modules. These routines have been developed on the basis of algorithms and recommendations, previously presented by Brage and Froese Fischer [49] and by Qui and Froese Fischer [64]. We follow closely their notation.

12.4.1. Evaluation of two-electron integrals by direct differential equation method

The two-electron radial integrals in atomic structure calculations are of the form

$$I^{k,p}(\rho,\sigma;\rho',\sigma') = \int_0^\infty\int_0^\infty f_\rho(r_1) f_\sigma(r_2)\frac{r_<^k}{r_>^{k+p}} f_{\rho'}(r_1) f_{\sigma'}(r_2)\,\mathrm{d}r_1\,\mathrm{d}r_2, \tag{12.12}$$

where $r_<$ and $r_>$ are the smaller and larger of the two coordinates r_1 and r_2, respectively. The $f(r)$ functions are linked to the radial orbitals, p is a positive integer, and k can assume a range of different integer values. If we define a function $Y^{k,p}(\sigma,\sigma';r)$ so that

$$I^k(\rho,\sigma;\rho',\sigma') = \int_0^\infty f_\rho(r)\frac{Y^{k,p}(\sigma\sigma';r)}{r^p} f_{\rho'}(r)\,\mathrm{d}r.$$

It follows that

$$Y^{k,p}(\sigma\sigma',r) = \int_0^r f_\sigma(s)\frac{s^k}{r^k} f_{\sigma'}(s)\,\mathrm{d}s + \int_r^\infty f_\sigma(s)\frac{r^{k+p}}{s^{k+p}} f_{\sigma'}(s)\,\mathrm{d}s. \tag{12.13}$$

It can be shown that this function satisfies a differential equation of the form

$$\begin{aligned}&\left(\frac{\mathrm{d}}{\mathrm{d}r} - \frac{k+p}{r}\right)Y^{k,p}(\sigma\sigma';r) = -\frac{2k+p}{r}Z^k(\sigma\sigma';r),\\ &Y^k(\sigma\sigma';r) \to Z^k(\sigma\sigma';r) \quad \text{as } r\to\infty.\end{aligned} \tag{12.14}$$

Here we have introduced a second function

$$Z^k(\sigma\sigma';r) = \int_0^r f_\sigma(s)\frac{s^k}{r^k} f_{\sigma'}(s), \tag{12.15}$$

which satisfies the differential equation

$$\left(\frac{\mathrm{d}}{\mathrm{d}r} - \frac{k}{r}\right)Z^k(\sigma\sigma';r) = f_\sigma(r) f_{\sigma'}(r); \qquad Z^k(\sigma\sigma';0) = 0. \tag{12.16}$$

The form of the Z^k function is quite general when defined in this way, since it is independent of p.

In Breit–Pauli calculations, all two-electron integrals can be reduced to four main types: R^k, N^k, V^k, and T^k integrals.

R^k integrals

$$R^k(\rho,\sigma;\rho',\sigma') = \int_0^\infty\int_0^\infty P_\rho(r_1) P_\sigma(r_2)\frac{r_<^k}{r_>^{k+1}} P_{\rho'}(r_1) P_{\sigma'}(r_2)\,\mathrm{d}r_1\,\mathrm{d}r_2. \tag{12.17}$$

This is the special case of Eq. (12.12) with $p=1$, and the f functions being the radial orbitals. The R^k integrals are highly symmetric

$$R^k(\rho,\sigma;\rho',\sigma') = R^k(\rho',\sigma;\rho,\sigma') = R^k(\rho,\sigma';\rho',\sigma) = R^k(\sigma,\rho;\sigma',\rho'),$$

i.e. they are unchanged as long as the ρ and ρ' orbitals, and thereby the σ and σ', are linked together with one coordinate.

The two-step procedure described above can be avoided, since the differential equation for the Y^k functions is

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}r^2} - \frac{(k+p)^2-(k+p)}{r^2}\right)Y^k(\sigma\sigma';r) = -\frac{(2k+p)(p-1)}{r^2}Z^k(\sigma\sigma';r) - \frac{2k+p}{r}f_\sigma(r) f_{\sigma'}(r). \tag{12.18}$$

In the special case of $p = 1$, the coefficient of the Z^k function disappears, and we are left with a single, though second-order, differential equation for Y^k

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}r^2} - \frac{k(k+1)}{r^2}\right) Y^k(\sigma\sigma'; r) = -\frac{2k+1}{r} f_\sigma(r) f_{\sigma'}(r) \tag{12.19}$$

with the boundary conditions

$$\begin{aligned} &Y^k_{\sigma\sigma'}(0) = 0, \\ &Y^k_{\sigma\sigma'}(r_{\max}) = \int_0^{r_{\max}} f_\sigma(r) f_{\sigma'}(r)\,\mathrm{d}r \quad \text{if } k = 0, \\ &\frac{\mathrm{d}}{\mathrm{d}r} Y^k_{\sigma\sigma'}(r_{\max}) - \frac{k}{r} Y^k_{\sigma\sigma'}(r_{\max}) = f_\sigma(r_{\max}) f_{\sigma'}(r_{\max}) \quad \text{if } k > 0. \end{aligned} \tag{12.20}$$

The boundary condition at $r_{\max}$ can be obtained from (12.13), assuming that the orbitals disappear at $r > r_{\max}$.

N^k integrals

$$\begin{aligned} N^k(\rho, \sigma; \rho', \sigma') &= \frac{\alpha^2}{4} \int_0^\infty \int_0^\infty P_\rho(r_1) P_\sigma(r_2) \frac{r_2^k}{r_1^{k+3}} \varepsilon(r_1 - r_2) P_{\rho'}(r_1) P_{\sigma'}(r_2)\,\mathrm{d}r_1\,\mathrm{d}r_2 \\ &= \frac{\alpha^2}{4} \int_0^\infty P_\rho(r) P_{\rho'}(r) \frac{1}{r^3} Z^k(P_\sigma, P_{\sigma'}; r)\,\mathrm{d}r \end{aligned} \tag{12.21}$$

where $k \geqslant -1$ and $\varepsilon(x) = 0$ for $x \leqslant 0$, $\varepsilon(x) = 1$ for $x > 0$. The $\varepsilon(x)$ step function reduces the Y^k function to Z^k, and the N^k integral is the special case of Eq. (12.12) with $p = 3$, $Y^k = Z^k$ and the f functions as the radial orbitals. Hence, in this case we need to solve only one first-order differential equation (12.16). But an additional complication occurs when dealing with the case N^{-1}, i.e. $k < 0$. In this case the homogeneous solution to Eq. (12.16) is

$$Z^k(\sigma\sigma', r) = \text{const} \times r^{-k}$$

which leads to an ambiguous result when $r \to 0$, since $k > 0$. This is solved by imposing a stronger boundary condition

$$\frac{\mathrm{d}}{\mathrm{d}r} Z^k(\sigma\sigma', 0) = 0 \quad \text{if } k < 0. \tag{12.22}$$

The N^k integrals have fewer symmetries than the R^k integrals:

$$N^k(\rho, \sigma; \rho', \sigma') = N^k(\rho', \sigma; \rho, \sigma') = N^k(\rho, \sigma'; \rho', \sigma).$$

In applications, we also use the integrals

$$M^k(\rho, \sigma; \rho', \sigma') = \bar{N}^k(\rho, \sigma; \rho', \sigma') = \frac{1}{2}\left[N^k(\rho, \sigma; \rho', \sigma') + N^k(\sigma, \rho; \sigma', \rho')\right] \tag{12.23}$$

which are as symmetric as the R^k integral and differ from the latter only by $p = 3$.

V^k integrals

$$V^k(\rho, \sigma; \rho', \sigma') = \frac{\alpha^2}{4} \int_0^\infty \int_0^\infty P_\rho(r_1) P_\sigma(r_2) \frac{r_<^k}{r_>^{k+3}} \left(\frac{\mathrm{d}}{\mathrm{d}r_1} - \frac{1}{r_1}\right) P_{\rho'}(r_1) r_2 P_{\sigma'}(r_2)\,\mathrm{d}r_1\,\mathrm{d}r_2. \tag{12.24}$$

This is the special case of Eq. (12.12) with $p = 3$ and

$$\begin{aligned} &f_\rho(r_1) = P_\rho(r_1); \qquad && f_{\rho'}(r_1) = \left(\frac{\mathrm{d}}{\mathrm{d}r_1} - \frac{1}{r_1}\right) P_{\rho'}(r_1); \\ &f_\sigma(r_2) = P_\sigma(r_2); \qquad && f_{\sigma'}(r_2) = r_2 P_{\sigma'}(r_2); \end{aligned}$$

The V^k integral allows only the two-step procedure (12.14)–(12.16), and it has the most restricted symmetry relations:

$$V^k(\rho, \sigma; \rho', \sigma') = V^k(\rho, \sigma'; \rho', \sigma).$$

In applications, we also use more the symmetric integrals [25]:

$$W^k(\rho,\sigma;\rho',\sigma') = \bar{V}^k(\rho,\sigma;\rho',\sigma') \\ = 2V^k(\rho,\sigma;\rho',\sigma') - (k+1)N^{k+1}(\rho,\sigma;\rho',\sigma') + (k+2)N^{k-1}(\sigma,\rho;\sigma',\rho'), \tag{12.25}$$

$$Q^k(\rho,\sigma;\rho',\sigma') = \bar{\bar{V}}^k(\rho,\sigma;\rho',\sigma') = \bar{V}^k(\rho,\sigma;\rho',\sigma') - \bar{V}^k(\sigma,\rho;,\sigma'\rho'). \tag{12.26}$$

T^k integrals

$$T^k(\rho,\sigma;\rho',\sigma') = \frac{\alpha^2}{4(2k+1)} \int_0^\infty \int_0^\infty P_\rho(r_1) P_\sigma(r_2) \frac{r_<^k}{r_>^{k+1}} \left(\frac{\partial}{\partial r_1} + \frac{1}{r_1}\right) P_{\rho'}(r_1) \left(\frac{\partial}{\partial r_2} + \frac{1}{r_2}\right) P_{\sigma'}(r_2)\, \mathrm{d}r_1\, \mathrm{d}r_2 \tag{12.27}$$

with $p = 1$ and

$$f_\rho(r_1) = P_\rho(r_1); \qquad f_{\rho'}(r_1) = \left(\frac{\mathrm{d}}{\mathrm{d}r_1} - \frac{1}{r_1}\right) P_{\rho'}(r_1);$$
$$f_\sigma(r_2) = P_\sigma(r_2); \qquad f_{\sigma'}(r_2) = \left(\frac{\mathrm{d}}{\mathrm{d}r_2} - \frac{1}{r_2}\right) P_{\sigma'}(r_2).$$

Here we can use the same procedure as for the R^k integrals, but the symmetry relations are stricter:

$$T^k(\rho,\sigma;\rho',\sigma') = T^k(\sigma,\rho;\sigma';\rho').$$

In the Breit–Pauli approximation, one also frequently uses the U^k integrals [25]. In the present implementation we do not use any special procedure for U^k, but instead the relationship

$$U^k(i,j;i',j') = T^k(i,j;i',j') + T^k(i',j;i,j'). \tag{12.28}$$

List of routines

spline_slater	module, contains common arrays used in the evaluation of two-electron integrals by the direct differential equation method.
bzk	computes the spline solution for the Z^k function.
facdyk	sets up and factorizes the matrix of the Y^k operator $\mathrm{d}^2/\mathrm{d}r^2 - k(k+1)/r^2$ in the B-spline basis.
facdzk	sets up and factorizes the matrix of the Z^k operator $\mathrm{d}/\mathrm{d}r + k/r$.
rky	evaluates the R^k integral by the direct differential method.
nky	evaluates the N^k integral by the direct differential method.
mky	evaluates the M^k integral by the direct differential method.
tky	evaluates the T^k integral by the direct differential method.
vky	evaluates the V^k integral by the direct differential method.
ykf	computes the spline solution for the Y^k function.

MODULE spline_slater

kz	multipole index for the Z^k function.
dzk`(1:ktx,ns)` **ipvtz**`(1:ns)` **ktx**`= 3*ks-2`	factorization of the matrix for the Z^k operator $\mathrm{d}/\mathrm{d}r + k/r$ in the B-spline basis.
ky	multipole index for the Y^k function.
dyk`(1:ktx,1:ns)` **ipvtd**`(1:ns)`	factorization of the matrix for the Y^k operator $\mathrm{d}^2/\mathrm{d}r^2 - k(k+1)/r^2$ in the B-spline basis.
yk`(1:ns)`	B-spline representation of the Y^k (or Z^k) function.

fyk`(1:nv,1:ks)`	values of Y^k (or Z^k) function at the Gaussian points.
fc1`(1:nv,1:ks)`, **ic1** **fc2**`(1:nv,1:ks)`, **ic2** **fy1**`(1:nv,1:ks)`, **iy1** **fy2**`(1:nv,1:ks)`, **iy2**	values at the Gaussian points and pointers for four one-electron radial functions in the two-electron integral under consideration.
fc`(1:nv,1:ks)`	work array.

Internal routines:

Allocate_slater	allocates space for all arrays according to parameters from the module *spline_param*.
Dealloc_slater	deallocates space in the module *spline_slater*.

Remarks.

- The routines **ykf**, **bzk**, **ytk**, and **yvk** differ only by their way of calculating the Y^k functions. The specific routines **ytk** and **yvk** are incorporated in the calling routines.
- The Y^k and T^k integrals are defined through the solution of second-order differential equations. The N^k integral needs only the Z^k solution of the first-order equation while the V^k integral is evaluated by the sequential solution of two first-order differential equations.
- The **ykf** routine includes the relativistic shift modification of the Y^k function if the parameter **rel** = 'TRUE'.
- The **quadr** routine is used to determine the appropriate boundary conditions (12.20) when $k = 0$. It is important in the case of non-orthogonal orbitals.
- These routines are very fast but not as accurate as the cell integration. They can be used for evaluating two-electron integrals, but not for generating the interaction matrix in the B-spline basis.

Calling sequences

```
        rky                                    nky
   ---------------                         ----------
  /              \                        /         \
 ykf            yval                    bzk        yval
  |                                      |
  -------------------------             -------------
  |     |       |     |   \             /       |    \
facdyk yval   vinty  quadr dgbsl      facdzk   yval   dgbsl
   |                                    |
dgbfa                                 dgbfa


              tky                             vky
          ----------                      ------------
        //          \                    //            \
      ytk           yval                yvk            yval
       |                                 |
    --------------------------       -------------------------
    |       |     |    |      \      |     |      |      |    \

  facdyk   yval vinty  quadr dgbsl   bzk  yval  vinty  facdzk dgbsl
   |                                  |                  |
  dgbfa                               --------------    dgbfa
                                    facdzk  yval  dgbsl
                                       |
                                     dgbfa
```

12.4.2. Evaluation of two-electron integrals in the B-spline basis

In the B-spline representation of radial atomic orbitals (12.5), the two-electron integrals (12.12) become

$$I^k(a,b;c,d)=\sum_i\sum_j\sum_{i'}\sum_{j'}a_ib_ic_{i'}d_{j'}I^k(i,j;i',j'), \tag{12.29}$$

where the matrix elements $I^k(i,j;i',j')$ are

$$\begin{aligned}
R^k(i,j;i',j')&=\int_0^\infty\int_0^\infty B_i(r_1)B_j(r_2)\frac{r_<^k}{r_>^{k+1}}B_{i'}(r_1)B_{j'}(r_2)\,\mathrm{d}r_1\,\mathrm{d}r_2,\\
N^k(i,j;i',j')&=\int_0^\infty \mathrm{d}r_1\,B_{i'}(r_1)B_i(r_1)\frac{1}{r_1^3}\int_0^{r_1}\mathrm{d}r_2\,B_j(r_2)\frac{r_2^k}{r_1^k}B_{j'}(r_2),\\
T^k(i,j;i',j')&=\int_0^\infty\int_0^\infty B_i(r_1)B_j(r_2)\frac{r_<^k}{r_>^{k+1}}\bar{B}_{i'}(r_1)\bar{B}_{j'}(r_2)\,\mathrm{d}r_1\,\mathrm{d}r_2,\\
V^k(i,j;i',j')&=\int_0^\infty\int_0^\infty B_i(r_1)B_j(r_2)\frac{r_<^k}{r_>^{k+3}}\bar{B}_{i'}(r_1)B_{j'}(r_2)r_2\,\mathrm{d}r_1\,\mathrm{d}r_2,
\end{aligned} \tag{12.30}$$

with $\bar{B}(r)=B'(r)-(1/r)B(r)$. The matrix elements (12.30) can be obtained by using the same procedures described above in Section 12.4.1 after replacing the $f(x)$ functions by the corresponding B-splines.

Therefore, the two-electron integral is given as a four-fold summation, but considerable simplification is possible. Note that

$$I^k(i,j;i'j')=0\quad\text{if either }|i'-i|\geqslant k_s\text{ or }|j'-j|\geqslant k_s. \tag{12.31}$$

Furthermore, the integrals may be symmetric relative to the (i,i') or (j,j') indexes. As a result, the B-spline integrals can be stored as a matrix with band character relative to the (i,i') or (j,j') indexes, namely:

$$\begin{aligned}
&R^k(n_s,n_s,k_s,k_s),\\
&N^k(n_s,n_s,k_s,k_s),\\
&V^k(n_s,n_s,2k_s-1,k_s),\\
&T^k(n_s,n_s,2k_s-1,2k_s-1).
\end{aligned} \tag{12.32}$$

In particular, the array element $R^k(i,j;i_p,j_p)$ refers to the matrix element in which $i'=i+i_p-1$ and $j'=j+j_p-1$, where $i_p,j_p=1,\dots,k_s$ (the symmetric upper-column mode). However, the array element $T^k(i,j;i_p,j_p)$ refers to the matrix element in which $i'=i+i_p-k_s$ and $j'=j+j_p-k_s$, where $i_p,j_p=1,\dots,2k_s-1$ (the non-symmetric column mode).

Thus, it is convenient to regroup the summation as follows:

$$R^k(a,b;c,d)=\sum_i\sum_{i'}a_ic_{i'}\left\{\sum_j\sum_{j'}b_jd_{j'}R^k(i,j;i',j')\right\}. \tag{12.33}$$

Let us consider the innermost double summation. Only the matrix elements with $j'\geqslant j$ are stored, but the summation in Eq. (12.33) runs over all values. However, the summation may be written as

$$\sum_j\sum_{j'}b_jd_{j'}R^k(i,j;i',j')=\sum_j b_jd_jR^k(i,j;i',j)+\sum_j\sum_{j'>j}(b_jd_{j'}+b_{j'}d_j)R^k(i,j;i',j'). \tag{12.34}$$

Let us define a direct "density matrix"

$$B(j,j_p)=\begin{cases}b_jd_j & \text{if } j_p=1,\\ b_jd_{j+j_p-1}+d_jb_{j+j_p-1} & \text{if } j_p>1\text{ and } j+j_p-1\leqslant n_s.\end{cases} \tag{12.35}$$

With a similar definition for the another pair of indexes,

$$A(i,i_p)=\begin{cases}a_ic_i & \text{if } i_p=1,\\ a_ic_{i+i_p-1}+c_ia_{i+i_p-1} & \text{if } i_p>1\text{ and } i+i_p-1\leqslant n_s,\end{cases} \tag{12.36}$$

the Slater integral is then defined as

$$R^k(a,b;c,d)=\sum_{i=1}^{n_s}\sum_{i_p=1}^{k_s}A(i,i_p)\sum_{j=1}^{n_s}\sum_{j_p=1}^{k_s}B(j,j_p)R^k(i,j;i_p,j_p). \tag{12.37}$$

Similarly, we can obtain

$$T^k(a,b;c,d)=\sum_{i=1}^{n_s}\sum_{i_p=1}^{2k_s-1}\bar{A}(i,i_p)\sum_{j=1}^{n_s}\sum_{j_p=1}^{2k_s-1}\bar{B}(j,j_p)T^k(i,j;i_p,j_p), \tag{12.38}$$

where

$$\begin{aligned}\bar{A}(i,i_p)&=a_i c_{i+i_p-k_s}, & 1<i+i_p-k_s<n_s,\\ \bar{B}(j,j_p)&=b_j d_{j+j_p-k_s}, & 1<j+j_p-k_s<n_s.\end{aligned} \tag{12.39}$$

The N^k integrals are also defined through the 'symmetric' summation (12.37), whereas the V^k integrals are the mixed case

$$V^k(a,b;c,d)=\sum_{i=1}^{n_s}\sum_{i_p=1}^{2k_s-1}\bar{A}(i,i_p)\sum_{j=1}^{n_s}\sum_{j_p=1}^{k_s}B(j,j_p)V^k(i,j;i_p,j_p). \tag{12.40}$$

List of routines

spline_integrals	module, contains the B-spline representation of the given two-electron integral.
convol	convolutes the four-dimensional array of B-spline two-electron spline integrals with a two-dimensional density matrix.
density	provides the density matrices (12.35) or (12.39).
mrk_diff	sets up the matrix for the B-spline representation of the R^k-integral using the differential equation method.
mnk_diff	sets up the matrix for the B-spline representation of the N^k-integral using the differential equation method.
mmk_diff	sets up the matrix for the B-spline representation of the M^k-integral using the differential equation method.
mtk_diff	sets up the matrix for the B-spline representation of the T^k-integral using the differential equation method.
mvk_diff	sets up the matrix for the B-spline representation of the V^k-integral using the differential equation method.
rk	evaluates the R^k integral by assembling the B-spline integrals.
nk	evaluates the N^k integral by assembling the B-spline integrals.
mk	evaluates the M^k integral by assembling the B-spline integrals.
tk	evaluates the T^k integral by assembling the B-spline integrals.
vk	evaluates the V^k integral by assembling the B-spline integrals.

MODULE spline_integrals

krk	multipole index for the integral under consideration.
itype	character `(rk, nk, mk, tk, or vk)` which indicates the type of integral.
rkb	the B-spline representation of the given two-electron integral, stored in column mode with dimensions (1:n_s,1:n_s,1:2k_s-1,1:2k_s-1).

Internal routines:

Allocate_integrals	allocates space for the `rkb` array according to parameters from module **spline_param**.
Dealloc_integrals	deallocates space in module **spline_integral**.

Calling sequences

```
                         rk                         nk

                          |                          |

                       mrk_diff                  mnk_diff

                         |                          |
                  -----------------          -----------------
                 /     ||     |    \        /       ||    |    \

            facdyk   bspyk   yval  minty   facdzk  bspzk  yval  minty
              |        |                     |       |
            dgbfa    dgbsl                  dgbfa   dgbsl


                            tk                          vk
                            |                           |
                         mtk_diff                    mvk_diff
                            |                           |
                     -----------------          -----------------
                    /      ||    |    \        /      ||     |   \

               facdyk    bsptyk  yval dinty   facdzk  bspvyk yval dinty
                 |          |                   |        |
               dgbfa       dgbsl              dgbfa ----------
                                                   /    |     \
                                                 yval  vinty  dgbsl
```

Remarks.

- The module **spline_integrals** contains only one array for all types of integrals. If requirements on memory are not very strict, it is possible to introduce specific arrays (with different shape) for each type of integrals.
- The **rk**, **nk**, **mk**, **vk**, and **tk** routines evaluate the two-electron integrals by assembling the matrix elements in the B-spline basis with the corresponding orbital expansion coefficients (see Eqs. (12.33)–(12.40)). The calculations of matrix elements in the B-spline basis are performed by the routines **mrk_diff**, **mnk_diff**, **mmk_diff**, **mvk_diff**, and **mtk_diff** by the differential equation method, or with the routines **mrk_cell**, **mnk_cell**, **mmk_cell**, **mvk_cell**, and **mtk_cell** by the cell integration method (see next section). All these routines first check the parameters `krk` and `itype` in the module *spline_integrals*. Only if these parameters differ from what is required, the calculation of the corresponding B-spline representation for the given integrals is initialized.
- The above routines differ mainly by their way of calculating the corresponding Z^k and Y^k functions between the individual B-spline functions in the **bspyk**, **bspzk**, **bsptyk**, and **bspvyk** routines, respectively. These specific routines are incorporated in the calling routines, and the algorithms are similar to the algorithms in routines **ykf**, **bzk**, **ytk**, and **yvk**, based upon the direct integration of two-electron integrals (see Section 12.3.2).
- When the parameter **rel** = 'TRUE', the **bspyk** routine includes the relativistic shift modification for the Y^k function due to the two-electron Darvin and spin–spin contact terms (see Section 2.6). The fine-structure multiplicative factor $\alpha^2/4$ is applied directly to the `rkb` array, if it contains a B-spline representation for the relativistic integral.

12.4.3. Cell algorithm for the evaluation of two-electron integrals

In this algorithm, the two dimensional region (r_1, r_2) is divided into a number of rectangular cells according to the chosen grid. The direct two-dimensional integration using Gaussian quadrature is performed for the evaluation of integrals within each cell. Over the off-diagonal cells, the integration is separable, so the two-dimensional cell integral is reduced to a product of two one-dimensional integrals. Furthermore, the scaling invariance of the B-splines in the logarithmic region of the chosen grid is fully exploited, and only a restricted amount of cell integrations need to be performed. The values of the required integrals can then be obtained by assembling the cell integrals. This algorithm significantly improves the efficiency and accuracy of the traditional methods that rely on the solution of the differential equations.

Integration over the off-diagonal cells. In this case the integrand in the matrix elements is separable and the two-dimensional integrals are reduced to a product of two one-dimensional integrals. Consider, for example, the Slater integral and assume $i_v < j_v$, where i_v and j_v enumerate the knot subintervals $[r_{i_v}, r_{i_v+1}]$ and $[r_{j_v}, r_{j_v+1}]$. Then

$$\begin{aligned}
R^k(i,j;i',j';i_v,j_v) &= \int_{r_{j_v}}^{r_{j_v+1}} \int_{r_{i_v}}^{r_{i_v+1}} \frac{r_<^k}{r_>^{k+1}} B_i(r_1)B_j(r_2)B_{i'}(r_1)B_{j'}(r_2)\,\mathrm{d}r_1\,\mathrm{d}r_2 \\
&= \int_{r_{i_v}}^{r_{i_v+1}} r_1^k B_i(r_1)B_{i'}(r_1)\,\mathrm{d}r_1 \int_{r_{j_v}}^{r_{j_v+1}} \frac{1}{r_2^{k+1}} B_j(r_2)B_{j'}(r_2)\,\mathrm{d}r_2 \\
&= r^k(i,i';i_v) \times r^{-(k+1)}(j,j';j_v),
\end{aligned} \tag{12.41}$$

where

$$r^m(i,i',i_v) = \int_{r_{i_v}}^{r_{i_v+1}} r^m B_i(r)B_{i'}(r)\,\mathrm{d}r. \tag{12.42}$$

The $r^m(i,i';i_v)$ quantities are called the integral moments. Although there are $n_v(n_v-1)$ off-diagonal cells, we only need to calculate and store the n_v two-dimensional moment arrays $r^m(i,i';i_v)$. Recall that each B-spline is a positive polynomial over a finite range. Hence, the moment integrals can be effectively evaluated using Gaussian quadrature with k_s Gaussian points. For a given order k_s, the spline B_i is nonzero only in the range from knot i to knot $i+k_s$, i.e. for a given cell i_v, the i and j indexes only run over the k_s values $i_v,\dots,i_v+k_s-1$. Hence, the moment arrays $r^m(i,i';i_v)$ have dimension (1:k_s;1:k_s,1:n_v). Furthermore, since $r^m(i,i';i_v)$ is symmetric with respect to interchanging the indexes i, i', only $r^m(i,i';i_v)$ with $i \leqslant i'$ needs to be evaluated. The evaluation can be further reduced by using their scaling properties in the logarithmic grid region. Suppose that the splines lie entirely within the exponential region. Let the left-most knot defining $B_i(r)$ be t_i and let $r = t_i + s$. Then

$$B_i(t_i+s) = B_{i+1}\big((1+h)(t_i+s)\big) = B_{i+1}\big((t_{i+1}+s(1+h))\big). \tag{12.43}$$

Based on this displacement invariance, the following scaling law holds for the moment integrals:

$$r^m(i+1,i'+1;i_v+1) = (1+h)^{1+m} r^m(i,i';i_v). \tag{12.44}$$

Expressions similar to (12.41) can also be written for the N^k, V^k, and T^k integrals:

$$\begin{aligned}
N^k(i,j;i',j';i_v,j_v) &= r^{-(k+3)}(i,i';i_v) \times r^k(j,j';j_v), && i_v > j_v, \\
T^k(i,j;i',j';i_v,j_v) &= d^{-(k+1)}(i,i';i_v) \times d^k(j,j';j_v), && i_v > j_v, \\
T^k(i,j;i',j';i_v,j_v) &= d^k(i,i';i_v) \times d^{-(k+1)}(j,j';j_v), && j_v > i_v, \\
V^k(i,j;i',j';i_v,j_v) &= d^{-(k+3)}(i,i';i_v) \times r^{k+1}(j,j';j_v), && i_v > j_v, \\
V^k(i,j;i',j';i_v,j_v) &= d^k(i,i';i_v) \times r^{-(k+2)}(j,j';j_v), && j_v > i_v,
\end{aligned} \tag{12.45}$$

where we introduced the moment integrals $d^m(i,i',i_v)$ containing derivatives:

$$d^m(i,i',i_v) = \int_{r_{i_v}}^{r_{i_v+1}} r^m B_i(r)\left(B'_{i'}(r) - \frac{1}{r}B_{i'}(r)\right)\mathrm{d}r. \tag{12.46}$$

Integration over the diagonal cells. In the diagonal cells, the two coordinates in the integrand of the two-electron integrals are coupled. Moreover, the integrand is not continuous across the cell diagonal where $r_1 = r_2$. We therefore separate the rectangular cell into two triangles, so that the integrand inside each triangle is continuous. The integration over the rectangular cell is then a summation of integrations over the two triangles, which are symmetric with respect to index exchange:

$$R^k(i,j;i',j';i_v) = R^k_\Delta(i,j;i',j';i_v) + R^k_\Delta(j;i;j',i';i_v), \tag{12.47}$$

where

$$R^k_\Delta(i,j;i',j';i_v) = \int_{r_{i_v}}^{r_{i_v+1}} \frac{1}{r_1^{k+1}} B_i(r_1)B_{i'}(r_1)\,\mathrm{d}r_1 \int_{r_{i_v}}^{r_1} r_2^k B_j(r_2)B_{j'}(r_2)\,\mathrm{d}r_2. \tag{12.48}$$

We use again Gaussian quadrature for the integration over the triangular cells. Here, however, we should generate new B-splines at the Gaussian points for the interval $[r_{i_v}, r_1]$. To minimize the number of new B-spline values needed to perform the two-dimensional integration, we apply the original Gaussian points used in the evaluation of the one-dimensional integrals to the integration with respect to the coordinate r_1 (i.e., the first integral in (12.48)). In the integration over the coordinate r_2, we also use the k_s Gaussian points for each interval $[r_{i_v}, r_1]$, dynamically calculating the required B-spline values. With this two-dimensional grid, the difficulty of the cell integrals near the origin due to the singularity of the integrand is minimized and uniformly accurate results for all Slater integrals can be obtained. The full storage of the $R^k_\Delta(i,j;i',j';i_v)$ integrals takes $n_v \times k_s^4$ locations ignoring the symmetry over the indexes. The computational time for these integrals can also be reduced by using the scaling law in the logarithmic grid region [64]:

$$R^k_\Delta(i+1,j+1;i'+1,j'+1;i_v+1) = (1+h) \times R^k_\Delta(i,j;i',j';i_v). \tag{12.49}$$

For the N^k integrals we have

$$N^k(i,j;i',j';i_v) = N^k_\Delta(i,j;i',j';i_v),$$

where

$$N^k_\Delta(i,j;i',j';i_v) = \int_{r_{i_v}}^{r_{i_v+1}} \frac{1}{r_1^{k+3}} B_i(r_1)B_{i'}(r_1)\,\mathrm{d}r_1 \int_{r_{i_v}}^{r_1} r_2^k B_j(r_2)B_{j'}(r_2)\,\mathrm{d}r_2.$$

Then for the T^k integrals:

$$T^k(i,j;i',j';i_v) = T^k_\Delta(i,j;i',j';i_v) + T^k_\Delta(j;i;j',i';i_v),$$

where

$$T^k_\Delta(i,j;i',j';i_v) = \int_{r_{i_v}}^{r_{i_v+1}} \frac{1}{r_1^{k+1}} B_i(r_1)B_{i'}(r_1)\,\mathrm{d}r_1 \int_{r_{i_v}}^{r_1} r_2^k \bar{B}_j(r_2)\bar{B}_{j'}(r_2)\,\mathrm{d}r_2,$$

and, finally, the V^k integrals:

$$V^k(i,j;i',j';i_v) = V^k_\Delta(i,j;i',j';i_v),$$

where

$$\begin{aligned} V^k_\Delta(i,j;i',j';i_v) = {} & \int_{r_{i_v}}^{r_{i_v+1}} \frac{1}{r_1^{k+3}} B_i(r_1)\bar{B}_{i'}(r_1)\,\mathrm{d}r_1 \int_{r_{i_v}}^{r_1} r_2^{k+1} B_j(r_2)B_{j'}(r_2)\,\mathrm{d}r_2 \\ & + \int_{r_{i_v}}^{r_{i_v+1}} \frac{1}{r_1^{k+2}} B_j(r_1)B_{j'}(r_1)\,\mathrm{d}r_1 \int_{r_{i_v}}^{r_1} r_2^k B_i(r_2)\bar{B}_{i'}(r_2)\,\mathrm{d}r_2. \end{aligned}$$

For all relativistic integrals, the scaling law in the exponential region is the same, but it differs from that for the R^k case:

$$I^k_\Delta(i+1,j+1;i'+1,j'+1;i_v+1) = I^k_\Delta(i,j;i',j';i_v)/(1+h). \tag{12.50}$$

Now one can obtain the B-spline representation for any integral by summing the corresponding cell integrals over the relevant cells.

Direct summation of cell integrals. Consider the cell (i_v, j_v). Its contribution to the total $R^k(a,b;c,d)$ integral is

$$R^k(a,b;c,d;i_v,j_v) = \sum_{i=i_v}^{i_v+k_s-1} a_i \sum_{j=j_v}^{j_v+k_s-1} b_j \sum_{i'=i_v}^{i_v+k_s-1} c_{i'} \sum_{j'=j_v}^{j_v+k_s-1} d_{j'} R^k(i,j;i',j';i_v,j_v). \tag{12.51}$$

Using the formula for the cell integrals, we obtain

$$\begin{aligned} R^k(a,b;c,d;i_v<j_v) &= \left(\sum_{i=i_v}^{i_v+k_s-1} a_i \sum_{i'=i_v}^{i_v+k_s-1} c_{i'} r^k(i,i',i_v)\right)\left(\sum_{j=j_v}^{j_v+k_s-1} b_j \sum_{j'=j_v}^{j_v+k_s-1} d_{j'} r^{-(k+1)}(j,j';j_v)\right) \\ &= X^k(a,c;i_v) \times X^{-(k+1)}(b,d;j_v), \\ R^k(a,b;c,d;i_v>j_v) &= \left(\sum_{i=i_v}^{i_v+k_s-1} a_i \sum_{i'=i_v}^{i_v+k_s-1} c_{i'} r^{-(k+1)}(i,i',i_v)\right)\left(\sum_{j=j_v}^{j_v+k_s-1} b_j \sum_{j'=j_v}^{j_v+k_s-1} d_{j'} r^k(j,j';j_v)\right) \\ &= X^{-(k+1)}(a,c;i_v) \times X^k(b,d;j_v), \end{aligned}$$

where

$$X^m(x,y;i_v)=\sum_{i=i_v}^{i_v+k_s-1} x_i \sum_{i'=i_v}^{i_v+k_s-1} y_{i'} r^m(i,i',i_v)=\boldsymbol{x}\times\boldsymbol{r}^m(i_v)\times\boldsymbol{y}. \tag{12.52}$$

Hence, the off-diagonal cell integrals for the atomic orbitals can also be represented as a product of two quantities, which depend only on the cell index. Introducing the four vectors

$$\begin{aligned}
&A(i_v)=X^k(a,c;i_v),\\
&B(j_v)=X^{-(k+1)}(b,d;j_v),\\
&C(i_v)=X^{-(k+1)}(a,c;i_v),\\
&D(j_v)=X^k(b,d;j_v),
\end{aligned} \tag{12.53}$$

the cell representation of the total R^k integral has the form

$$R^k(a,b;c,d)=\left|\begin{matrix}
DIAG_1 & C_1D_2 & C_1D_3 & \vdots & C_1D_{n_v}\\
A_2B_1 & DIAG_2 & C_2D_3 & \vdots & C_2D_{n_v}\\
A_3B_1 & A_3B_2 & DIAG_3 & \vdots & C_3D_{n_v}\\
\cdots & \cdots & \cdots & \cdots & \cdots\\
A_{n_v}B_1 & A_{n_v}B_2 & A_{n_v}B_3 & \vdots & DIAG_{n_v}
\end{matrix}\right|. \tag{12.54}$$

Therefore, the R^k integral can be obtained by directly summing the above quantities, without generating the corresponding B-spline representation for the given integral described in the previous sections. Such procedure is much more efficient.

List of routines

spline_moments	module, contains the off-diagonals moments and diagonal-cell integrals for a given type of integral.
moments	evaluates the moments $\langle B_i\|r^\lambda\|B_j\rangle$ or $\langle B_i\|r^\lambda\|B'_j\rangle$ for a given power index.
rk_moments	generates all off-diagonals moments and diagonal-cell integrals for the R^k-integral.
nk_moments	generates all off-diagonals moments and diagonal-cell integrals for the N^k-integral.
mk_moments	generates all off-diagonals moments and diagonal-cell integrals for the M^k-integral.
tk_moments	generates all off-diagonals moments and diagonal-cell integrals for the T^k-integral.
vk_moments	generates all off-diagonals moments and diagonal-cell integrals for the V^k-integral.
mrk_cell	generates the B-spline representation of the R^k-integral using the cell-integration method.
mnk_cell	generates the B-spline representation of the N^k-integral using the cell-integration method.
mmk_cell	generates the B-spline representation of the M^k-integral using the cell-integration method.
mtk_cell	generates the B-spline representation of the T^k-integral using the cell-integration method.
mvk_cell	generates the B-spline representation of the V^k-integral using the cell-integration method.
rkc	evaluates the R^k integral by direct assembling the moments.
nkc	evaluates the N^k integral by direct assembling the moments.
mkc	evaluates the M^k integral by direct assembling the moments.
tkc	evaluates the T^k integral by direct assembling the moments.
vkc	evaluates the V^k integral by direct assembling the moments.

MODULE spline_moments. This module contains moments, and the two-electron integral over the diagonal cell, Eq. (12.49). These moments are used for calculating the two-electron integrals according to the cell algorithm.

kmk	multipole index for the integral under consideration.
mtype	character (rk, nk, mk, tk, or vk), which indicates the type of integral.
rkd (1: $k_s \times k_s$,1: $k_s \times k_s$,1: n_v)	stores the two-electron integrals between individual B-splines over the diagonal cells.
rkd1 (1: k_s,1: k_s,1: n_v) **rkd2** (1: k_s,1: k_s,1: n_v) **rkd3** (1: k_s,1: k_s,1: n_v) **rkd4** (1: k_s,1: k_s,1: n_v)	one-electron moments $r^m(i,i';i_v)$ and $d^m(i,i';i_v)$ as defined in Eqs. (12.44) and (12.46), which define the integration over the off-diagonal cells for a given type of integrals.

Internal routines:

Allocate_monents	allocates space for arrays according to parameters from module **spline_param**.
Dealloc_moments	deallocates space in module **spline_moments**.

Calling sequences

```
            rk_moments                         nk_moments
               |                                  |
              ----                               ----
             /   \\                             /   \\
       moments rk_pdiag                   moments nk_pdiag
                  ||                                 ||
             rk_triang                          nk_triang


       tk_moments                         vk_moments
            |                                  |
           ----                               ----
          /   \\                             /   \\
     moments tk_pdiag                   moments vk_pdiag
                ||                                 ||
          tk_triang                          vk_triang
```

Remarks.

- All the above routines for the calculations of the moments have the same structure and only differ by their way of calculating the diagonal cell contributions with the **triang**-routines. The **pdiag**-routines control the scaling properties.
- The **mrk_cell**, **mnk_cell**, **mmk_cell**, **mtk_cell** and **mvk_cell** routines generate the B-spline representation for the given two-electron integral and place the corresponding information in the array **rbk** in the module **spline_integral**.
- The direct cell-integration routines **rkc**, **nkc**, **mkc**, **tkc**, and **vkc** use the direct summation of elements in (12.54). They are the most effective and accurate way to calculate individual atomic integrals. However, **mrk_cell** and similar routines should be used to generate the interaction matrices.

12.5. The library ZCONF

This library contains subroutines and functions that deal with the configuration and close-coupling expansions. They were designed to unify the representation of different expansions and are used in almost all programs of the BSR package. The main modules in ZCONF are **configs**, **channels** and **target**, with handle data related to the configuration expansion, the scattering channels, or the description of the target states, respectively. The description of the atomic states in the present package is based upon the configuration model, and the corresponding expansions for each state should be represented in the so-called c-files, which are used in the MCHF atomic structure package [52]. For completeness of the description, we recall below the main features of the c-files.

A general configuration consists of groups of equivalent electrons

$$(n_1l_1)^{w_1}(n_2l_2)^{w_2}\dots(n_ml_m)^{w_m}, \qquad N=\sum_{i=1}^{m} w_i. \tag{12.55}$$

```
  C     3P          -37.7603504
 1s
 2s( 2)  2p( 2)                                   0.97657139
1S0 3P2 3P
 2p( 4)                                           -.13532612
 3P2
 2s( 1)  2p( 2)   3d( 1)                          0.10709232
2S1 1D2 2D1 2D   3P
 2s( 1)  2p( 2)   3d( 1)                          0.06814467
2S1 3P2 2D1 4P   3P
 2s( 1)  2p( 2)   3d( 1)                          -.06450278
2S1 3P2 2D1 2P   3P
 2s( 1)  2p( 1)   3s( 1)   3p( 1)                 0.05052374
2S1 2P1 2S1 2P1  3P  2P   3P
 2s( 2)  3p( 2)                                   -.03579417
1S0 3P2 3P
 2s( 2)  3d( 2)                                   0.03468992
1S0 3P2 3P
 2p( 2)  3d( 2)                                   -.03455937
3P2 1S0 3P
 2s( 1)  2p( 1)   3s( 1) 3p( 1)                   -.02901590
2S1 2P1 2S1 2P1  3P  2P 3P
 2p( 2)  3s( 2)                                   -.02546691
3P2 1S0 3P
*
```

Fig. 14. Structure of a c-file (representing the ground state of C as an example).

Each subshell is additionally characterized by its term $\alpha_i L_i S_i$. The total orbital angular momentum and the total spin are obtained by coupling the momenta of all occupied subshells according to the given scheme, completely specified by the intermediate angular momentum quantum numbers L_i', S_i'. In this package, the coupling is from left to right, for which the notation

$$\gamma LS = (n_1l_1)^{w_1}\alpha L_1S_1; (n_2l_2)^{w_1}\alpha L_2S_2, L_2'S_2'; (n_3l_3)^{w_1}\alpha L_3S_3, L_3'S_3'; \ldots (n_ml_m)^{w_m}\alpha L_mS_m, L_m'S_m'; LS \tag{12.56}$$

can be used. Here $|L_1 - L_2| \leqslant L_2' \leqslant |L_1 + L_2|$, $|L_3 - L_2'| \leqslant L_3' \leqslant |L_3 + L_2'|$, etc. Other coupling schemes may prove to be more advantageous in special circumstances. Therefore, each configuration state function, CSF, is defined by its configuration (12.55) and the coupling scheme (12.56). The c-file for a given atomic state contains the list of the CSFs involved, with the corresponding expansion coefficients as an option when the given c-file describes the specific atomic state. An example of a c-file is given in Fig. 14. All orbital angular momenta in the c-file are given in spectroscopic notation. The configuration states are specified by first listing the subshells and their occupation in a linear form where, for example, $2p^3$ becomes `2p(3)`, followed by a separate line with multiplicity $(2S+1)$, the coupled orbital angular momentum (L), and the seniority, first for each subshell, and then for the resultant when coupling these subshells from left to right. For the latter, the seniority is no longer relevant. Notice that for the same configuration there are several possible CSFs, which differ by the intermediate angular quantum numbers. For fine-structure LSJ atomic states, the c-file may also contain CSFs with different total terms. The expansion coefficients are given at the end of the configuration line. The first line of the c-file is reserved for the title (optional) and the energy of the given state. In the present implementation, the user can also define the J-value (if relevant) by including the statement `2J = value` at any position in the first line. The second line in the c-file contains the list of fully occupied, or closed, subshells common to all CSFs. This saves space and is also justified by the fact that the treatment of closed shells at the computation of matrix elements can be simplified by redefining the radial integrals for the outer electrons [68].

The spectroscopic notation adopted in the c-files is very convenient for the user, but it imposes several restrictions to the possible CSFs. First, the spectroscopic symbols for the angular momenta are restricted by the character map used in the computer. For values up to $l = 21$ we adopt the usual alphabetical characters $s, p, d, f, \ldots, z$, whereas for higher values (which can be important in scattering calculations), we use the symbols from the computer character map with indexes from 101 and higher. Note that the one-electron spectroscopic notation, in addition to the nl-values, may also contain the so-called 'set index', which indicates the non-orthogonal orbitals. For example, we can use the notation `3p1`, `3p2`, … for the term-dependent radial functions for the 3p orbital in the various $2p^5 3p$ LS states, or we can use different set indexes for orbitals obtained in independent calculations. The c-file provides only a four-character space for each orbital, and this imposes additional restrictions on the maximum value of possible principal quantum numbers and set indexes (for details, see the routines **al**, **elf**, **new_index**). However, in practical calculations these restrictions are usually not critical.

Note also that the c-file format was changed from its original presentation in the MCHF program [52,68]. Previously, it was assumed that there will be at most five subshells (filled or unfilled) outside the closed subshells common to all CSFs. This turned out to be a serious restriction, especially in scattering calculations, where we have an additional continuum orbital, i.e. an additional shell in the CSF. The format of the c-file was changed, so that we can now use CSFs with up to eight subshells outside the closed subshells.

In addition to the c-files, the present BSR package can also use the l- and j-files provided by the MCHF complex. The **name.l** file represents the list of solutions obtained in the frame of the configuration-interaction method on the basis of configurations in the file **name.c**. The **name.j** file corresponds to the list of solutions obtained in the frame of *LSJ* Breit–Pauli calculations.

Below we provide only the alphabetical list of modules and routines, with a brief description of the main points. The detailed description of the input/output parameters can be found at the beginning of the corresponding source codes.

Modules

configs	contains different information used for the description of the configuration expansions under consideration.
channel	defines the scattering channels for one partial wave.
channels	defines the scattering channels for all partial waves.
target	contains the description of the target states.

Routines

allocations	provides different array allocations in the modules.
decode_c	decodes the spectroscopic representation of the CFSs in the c-file to the ‘integer’ representation used by the programs.
det_exp	provides the determinant expansion for the given subshell.
gen_conf	generates all CFSs that can be built from a given configuration by adding the additional electron *nl*.
idef_jot	defines the total electronic angular momentum J for the configuration expansion in the given c-file.
idef_label	generates the list of labels for the CSFs in the given c-file.
idef_ncfg	defines the number of configurations in the given c-file.
idef_ne	defines the number of electrons for the CSFs in the given c-file.
idef_prt	defines the parity for the CSFs in the given c-file.
idef_st	defines the number of states in the given l- or j-file.
idef_term	defines the total term *LS* for the configuration expansion in the given c-file.
incode_c	incodes the ‘integer’ configuration representation used by the programs into spectroscopic c-file format.
new_index	assigns a new set index for the orbital on the base of the given orthogonal conditions.
pre_iort	prepares the orthogonality conditions for the one-electron radial functions according to their set indexes and additionally imposed conditions.
pri_conf	prints the given configuration in spectroscopic format.
r_closd	reads the list of common closed shells from the given c-file.
r_confc	reads the configuration list from the given c-file.
r_expn	reads the expansion coefficients from the given c-file.
r_orbit	defines the list of one-electron orbitals in the given c-file.
r_target	reads the information about the target states and the close-coupling expansion from the file **target**.
r_term	defines the list of different terms in the given c-file.
sym_conf	defines the configuration and angular symmetries for the given CSF.
test_a	checks the “AFTER” relations between the orbitals in the given configuration.
test_c	checks the configurations (number of electrons, parity, couplings).

12.6. The library ZCOM

This library contains subroutines and functions that are part of the BSR package, but they also provide a variety of auxiliary computations common to different atomic-structure calculations. This includes, for example, the calculation of 3j-symbols or various sorting procedures. The routines in ZCOM do not use any COMMON or USE Fortran statements, and all input and output parameters are transferred through the formal arguments only. No call is made from the library to any routine outside the library, with the exception of interface subroutines used for calling the LAPACK routines.

Below we provide the alphabetical list of routines, with only a brief description of the main points. The input/output parameters and the method used by each routine are fully commented at the beginning of the corresponding source code. If a user needs to use some routine in other applications than a BSR calculation, or wants to replace it with a more effective procedure, we advise to first read the relevant source code for a detailed description of formal arguments.

al	provides the spectroscopic symbol $(s, p, d, \ldots)$ for an orbital angular momentum l.
clebsh	determines a Clebsch–Gordan coefficient via the corresponding 3j-symbol.
conv_au	provides the transformation factor from a.u. to cm^{-1} or eV.
det	calculates the determinant value in the Gauss method.
dj_fact	defines the J-dependent factor for the reduced matrix elements of an electric multipole transition operator.
djm_fact	defines the J-dependent factor for the reduced matrix elements of a magnetic multipole transition operator.
el_nlk	decodes the spectroscopic notation for the electron orbital (n, l, k).
elf	provides the spectroscopic notation for the electron orbital (n, l, k).
f_gauss	gives the value of a given Gaussian at the point x.
gauss	looks up the values of the Gaussian coordinates and weights for a k-point Gaussian quadrature over the interval $[0, 1]$.
gen	provides the fractional parentage coefficients for p^q- and d^q-shells.
hwf	routines in this module provide hydrogenic bound radial functions and average values of $r^1, r^2, r^3, r^{-1}, r^{-2}, r^{-3}$ for a given nuclear charge Z.
iglq	total statistical weight of l^q-subshell.
interv	a reformatted version of the de Boor routine [60] with the same name.
inv	calculates the inverse matrix for the square input matrix `A(n,n)`.
ipointer	determines the position of an element i in the input integer array, with output value 0 if that element is not a member of the set.
iterm	provides information about possible atomic shell terms.
itri	checks the triangular relations for three coupled angular momenta.
la	gives the value of the orbital momentum l for a given spectroscopic symbol.
lap_call	contains interface routines for calling the most 'popular' LAPACK procedures, with the possibility of automatic allocation of work arrays if needed.
num	finds the integer representation for an arbitrary real number r, i.e. it finds such i_1 and i_2 that $\text{abs}(r) = \text{sqrt}(i_1/i_2)$ and $\text{sign}(r) = \text{sign}(i_1)$. It is used for the output of angular coefficients.
read_par	a set of routines for reading the parameter values, presented in the input file as *name* = *value*.
read_arg	a set of routines for reading the input arguments presented in the command line as *name* = *value*.
rttc	an interface program for timing routines. The user may choose the proper Fortran time routine that is most appropriate for a given computational platform.
t_ls_jk	provides the recoupling coefficients from *LS*- to *jj*-coupling.
t_ls_jk	provides the recoupling coefficients from *LS*- to *jK*-coupling.

xlagr	contains k points of a Lagrange interpolation procedure.
z_3j	calculates a 3j-symbol without directly using factorials.
z_6j	calculates a 6j-symbol without directly using factorials.
z_9j	calculates a 9j-symbol as a sum of 6j-symbols.
zcb	provides the square of a 3j-symbol with zero magnetic numbers, using a simplified algorithm.
zclkl	calculates the reduced matrix elements of spherical harmonics, $\langle l\|C^k\|l'\rangle$, without directly using factorials.

13. Additional programs-utilities

In addition to the programs that were described separately, a number of short utility programs are available that assist in the processing and managing of the data.

BOUND_BSW

Description:	converts the close-coupling B-spline expansions to the c- and *bsw*-files
Input files:	**bound.nnn, bound_bsw.inp**
Output files:	c- and *bsw*-files for given states
Call:	**bound_bsw < bound_bsw.inp**

The results of the bound-state B-spline calculations are recorded in the **bound.nnn** files in the form of the corresponding close-coupling expansions (7.3). They are similar to the l- or j-files in the MCHF complex, and contain descriptions of all states for a given total term. The BOUND_BSW utility collects the information for a given bound state indicated in the file **bound_bsw.inp**, and records it as pair **name.c** and **name.bsw**. These files can be used as input target files for the BSR scattering calculations.

BOUND_TAB

Description:	produces the total list of resulting bound states
Input files:	**bound.nnn**
Output files:	**bound.tab**
Call:	**bound_tab**

This program sorts the energy levels in different **bound.nnn** files and prints them in atomic units, eV or cm^{-1}, relative to the lowest state. The user can define the range of partial waves and the energy range to restrict the output.

BSW_W

Description:	converts the *bsw*-file to the w-file
Input files:	**name.bsw**
Output files:	**name.w**
Call:	**bsw_w name.bsw**

CFILE

Description:	extracts the given state from the l- or j-file in separate c-files
Input files:	**name.l** or **name.j**
Output files:	**result.c**
Call:	**cfile** `name.l(j) nn 2J result.c eps_c`

In the BSR complex, each target state is described by a pair **name.c** and **name.bsw**. The CI program from the MCHF complex produces the results in the $l(j)$-files as a list of states. The CFILE utility allows one to extract the expansion coefficients for a given state and create the corresponding c-file. The user should indicate in the command line the name of the input l- or j-file, the pointer for the given state, `nn`, the value of $2J$ if it is a j-file, the name for the output c-file, and the tolerance for the expansion coefficients, eps_c.

PHOTO_TAB

Description:	produces the tables for channel or total cross sections and asymmetry parameters
Input files:	**bsr_phot.nnn**
Output files:	indicated by user
Call:	**photo_tab** with interactive response

This utility serves for processing the data recorded during photoionization calculations with the program BSR_PHOT. The program BSR_PHOT can be run for different energy intervals and different partial waves separately. The results are accumulated in files **bsr_phot.nnn**. Then final tables for channel or total cross sections and asymmetry parameters may be generated with the utility **photo_tab**.

SLATER_W

Description:	transfers Slater-type orbitals to MCHF format
Input files:	**name.slw**
Output files:	**name.w**
Call:	**slater_w name.slw**

The Slater orbitals are supposed to be given as in the RMATRX1 code, i.e.

$$P_{nl}(r) = \sum_{j=1}^{\mathrm{NCO}} C(j) * r^{\mathrm{IRAD}(j)} * e^{-\mathrm{ZE}(j)*r}.$$

The formatted file name.slw contains the Slater-type orbital parameters in the form:

1.	`Z, atom, term`	nuclear charge, and optional identification for atom and term
2.	`nwf`	number of radial wave functions
3.	`n,l,k`	quantum numbers for given orbital
4.	`NCO`	
5.	`IRAD(1:NCO)`	
6.	`ZE(1:NCO)`	
7.	`C(1:NCO)`	

Repeat records 3–7 `nwf` times.

SUM_HH

Description:	merges the set of **h.nnn** files to the final **h.dat**
Input files:	**h.nnn**
Output files:	**h.dat**
Call:	**sum_hh** `klsp1 klsp2`

W_BSW

Description:	converts the w-file to the *bsw*-file format
Input files:	**name.w**
Output files:	**name.bsw**
Call:	**bsw_w name.w**

14. Compilation and test runs

This section describes the main features concerning the distribution, installation and compilation, and running of the system. Several test runs are presented in order to verify that the system works as expected.

14.1. Compilation

The description below and the script files are prepared for a UNIX/LINUX user in the *c*-shell. Other user can take these files as a guide for compilation in a specific environment.

The source-code files for the BSR complex are distributed in the compressed **bsr_cpc.tar.gz** file. In order to uncompress the source files, the user may employ the system routines

```
gzip -d bsr_cpc.tar.gz
tar -xvf bsr_cpc.tar.
```

All source files are unpacked into the directory BSR_CPC. Each main program, libraries or utilities are presented by a single source file. The instructions for compiling the entire system are given in the **make_BSR** script file. Before running **make_BSR**, the user should check the assignment of two additional environmental variables in this file. First the user should define the name of the FORTRAN compiler and optimization level in the variable `FC`. For example:

```
setenv FC ’fort -fast’.
```

Then the user should indicate in the variable `LAPACK` the directory where the linker can find the LAPACK and BLAS libraries. Recall that these popular libraries were chosen to provide the linear-algebra calculations in the BSR complex, and the user should install the LAPACK and BLAS libraries before compiling the BSR complex. The LAPACK and BLAS libraries can be obtained without charge from the web-site http://www.netlib.org/lapack/index.html. Many FORTRAN compilers also provide these libraries as an additional option. If the user copies the `liblapack.a` and `libblas.a` files to the BSR_CPC directory, then the variable LAPACK is defined as

```
setenv LAPACK ’-L. -llapack -lblas’.
```

Otherwise the corresponding path should be indicated after the parameter `-L`.

After setting the environment variables, the **make_BSR** file provides the recommended sequence of commands for compilation of the libraries and other BSR programs. The user can run this file as a command script file. As a result, all executable files for the main programs and utilities will be placed in the directory BSR_CPC. In order to use these programs in the test runs or in the production calculations, the user should make the corresponding modification in the shell variable `path`. For example, as a command

```
set path = ($HOME/BSR_CPC $path)
```

or `by putting`

```
setenv PATH ”$HOME/BSR_CPC: $PATH”
```

into the shell script file.

14.2. Test runs

We provide four test runs to illustrate the main modes of operation of the BSR complex, namely, *LS*-coupling scattering calculations, *JK*-coupling scattering calculations, photoionization, and bound-state calculations. In order to provide examples of realistic calculations and choices of appropriate physical models, the test runs were taken from published calculations for bound states [69] and electron scattering [70] from carbon atoms. Since these calculations were rather extensive, the test runs are given here in a simplified version. They only serve as a benchmark to ensure that the programs are running correctly, and as a template for users to construct their own runs.

Each scattering calculation begins with the choice and generation of the target states. In the framework of the BSR complex, this is the responsibility of the user. In principle, the user can employ any atomic-structure program, and then convert the results to the form acceptable in the BSR programs. The BSR programs themselves are adapted to use the target states from the MCHF complex of Froese Fischer [68]. Below we assume that all target states are generated in the framework of the MCHF approach, and we will show the run sequences in the different modes.

In the following test-run description we only discuss the input data files and the sequence of required commands. A complete set of input files and files with the final results is provided in the separate tar-file **bsr_test.tar.gz**, which contains the main directory BSR_TEST and four subdirectories C_LS, C_JK, C_PHOT, C_BOUND for each test run described below. Each subdirectory contains the file **run_script**, with a short description of the test and a required sequence of commands. Note that each target state is represented by a pair of c- and *frm*-files. The radial functions are given in the ASCII *frm*-file instead of unformatted w- or *bsw*-files, in order to simplify the test runs on different computer platforms, which may use a different default structure for unformatted files. As a consequence, the **run_script** file contains first a rather long sequence of preliminary commands for transformation of ASCII *frm*-files to the unformatted *bsw*-files required by the BSR complex. Each test-run subdirectory contains the file with the correct final results. These files have the extension *.test*. At the end of each **run_script** file we give information what files with final results should be compared in order to verify that the system works correctly.

14.3. LS-coupling test run

This test corresponds to low-energy elastic and inelastic electron scattering from carbon atoms. The close-coupling expansion includes the seven lowest states of neutral carbon with configurations $1s^22s^22p^2$, $1s^22s2p^3$, and $1s^22s^22p3s$. These states exhibit a strong term-dependence of the one-electron orbitals, and it is difficult to obtain accurate wavefunctions with only one orthogonal set of orbitals. In order to illustrate the advantage of the present complex, the atomic wavefunctions for odd-parity and even-parity states were generated using two different non-orthogonal sets of orbitals. This allows us to obtain fairly accurate target wavefunctions with relatively small configuration expansions. The results of the MCHF calculations are presented in the c- and w-files, for the configuration expansions and the one-electron radial functions, respectively. These files are the input for the present BSR calculations. The c-file for the carbon ground state is given in Section 12.5.

The first step of the BSR calculations concerns the choice of the B-spline knot sequence. The user should prepare the **knot.dat** file, indicating the R-matrix radius and the maximum size of subintervals, as discussed in Section 12.2 (see also Fig. 13). The w-files are then transformed to *bsw*-files with the program-utility **w_bsr** (see Section 13). At this stage the user can check the R-matrix radius by considering the value of the one-electron orbitals at the boundary. It is recommended that all orbitals should at least have fallen to magnitudes $< 10^{-3}$. Recall that the value of the R-matrix radius also automatically defines the number of B-splines. Since the time of the calculations is approximately proportional to the square of the number of B-splines, the user should choose the minimal possible R-matrix radius with a minimum number of B-splines.

The next step involves the preparation of the file **target**, with the specification of the scattering model, as shown below.

```
   e + C
--------------------------------------------------------------------------
coupling = LS      !   LS coupling
nz = 6             !   nuclear charge
nelc = 6           !   number of electrons
--------------------------------------------------------------------------
ntarg = 7          !   number of target states
--------------------------------------------------------------------------
2p2_3P.c
2p2_1D.c
2p2_1S.c
2p3_5So.c
2p3s_3Po.c
2p3s_1Po.c
2p3_3Do.c
--------------------------------------------------------------------------
nlsp =    12       !   number of partial waves
--------------------------------------------------------------------------
2Se     0    2     1   no
4Se     0    4     1   no
2So     0    2    -1   no
4So     0    4    -1   p_4So.c
2Pe     1    2     1   no
4Pe     1    4     1   no
2Po     1    2    -1   p_2Po.c
4Po     1    4    -1   no
2De     2    2     1   no
4De     2    4     1   no
```

```
2Do     2    2    -1   p_2Do.c
4Do     2    4    -1   no
--------------------------------------------------------------------------
```

The test run contains 12 partial waves, and for three partial waves there are additional c-files. These provide an example of $(N+1)$-electron configurations, which may be added to the close-coupling expansion to improve the description of short-range correlations. In the present example, the $(N+1)$-electron configurations were added for a more accurate description of the lowest resonance states with configuration $2p^3$. The orbitals in these files were optimized in separate MCHF calculations for each given term of the $2p^3$ configuration, and they are not orthogonal to the target orbitals. These additional multi-configuration expansions for the $2p^3$ states may be treated in two ways. First, they may be completely included in the close-coupling expansion, in which case we need to impose the orthogonality constraint for the corresponding continuum orbital according to $\langle k\mathrm{p}|3\mathrm{p}\rangle = 0$. Alternatively, we may remove from these perturber files the main configuration $2p^3$, leaving only the correlation configurations. We chose this way in the test run.

The last step in preparing the input data concerns the **bsr_par** file. Its parameters will control the specific run. This test run has only a few parameters that differ from the default values, namely:

```
# bsr_conf

<   2s|   ks>=0

#  bsr_hd:

itype = 0
idiag = 2
Emax = -0.1
```

The above orthogonality condition requires that all *ks* continuum orbitals should be orthogonal to the 2s orbital. By default, the continuum orbitals are not orthogonal to the bound target orbitals, but it is very important to keep orthogonality to the closed subshell as discussed in Section 5.1. Orthogonality to the closed core shells (here the 1s orbital) is imposed automatically. The orthogonality of the *ks* orbital to the 2s orbitals in the channel $2s^2 2p^2 ks$ is also imposed automatically by the BSR_CONF program. However, it is also important to keep the *ks* orbital orthogonal in all other channels (such as $2s2p^3 ks$); otherwise different channels may generate the same $(N+1)$-electron states ($2s^2 2p^3$ states in the present example). Note that no additional $(N+1)$-electron states are used in this case to compensate for the orthogonality conditions.

In this test we also chose the two-step diagonalization procedure (`idiag=2`, see Section 8.1). It is more effective, because it reduces the size of the interaction matrix to be diagonalized.

After preparing the target and input files, the run of the BSR programs is described by the following script file:

```
bsr_prep
bsr_conf
bsr_breit  klsp1=1  klsp2=12
bsr_mat    klsp1=1  klsp2=12
bsr_hd     klsp1=1  klsp2=12
sum_hh     1 12
```

The result of this run will be the H.DAT file, which may be used in subsequent cross-section calculations with some external-region program. If additional partial waves are needed (or we want to recalculate some partial waves), we may run additional calculations only for specific partial waves and then again the **sum_hh** utility to obtain the combined H.DAT file. The user may control the calculations using the information from the BSR_MAT and BSR_HD programs in the corresponding log-files. For example, the BSR_MAT program compares the input and the calculated target energies. If they do not agree within some tolerance, it means that the input c- or w-files contain errors. If the BSR_HD program fails to diagonalize the interaction matrix, this means that the total overlap matrix is not positively definite. This may indicate that not all important orthogonal constraints are imposed, or that the input target states are not orthogonal or not normalized. In order to verify that the system works correctly, user should only compare the final R-matrix eigenvalues given by the system in the file **sum_hh.log** and provided in file **sum_hh.test** for comparison. If these final eigenvalues do not agree with each other, we suggest to check all *log*-files for possible warnings.

14.4. jK-coupling test run

This test corresponds to low-energy elastic and inelastic scattering of electron by atomic carbon in the same approximation as the above *LS*-coupling test run, but with the inclusion of relativistic corrections. The **target** file now has the following structure:

```
    e + C
-----------------------------------------------------------------------
coupling = JK      !   coupling
nz = 6             !   nuclear charge
nelc = 6           !   number of electrons
-----------------------------------------------------------------------
ntarg = 13         !   number of target states
-----------------------------------------------------------------------
2p2_1D2.c
2p2_1S0.c
2p2_3P0.c
2p2_3P1.c
2p2_3P2.c
2p3s_1Po1.c
2p3s_3Po0.c
2p3s_3Po1.c
2p3s_3Po2.c
2p3_3Do1.c
2p3_3Do2.c
2p3_3Do3.c
2p3_5So2.c
-----------------------------------------------------------------------
nlsp = 12          !   number of partial waves
-----------------------------------------------------------------------
 1+    1     0     1   no
 1-    1     0    -1   p_1o.c
 3+    3     0     1   no
 3-    3     0    -1   p_3o.c
 5+    5     0     1   no
 5-    5     0    -1   p_5o.c
 7+    7     0     1   no

 7-    7     0    -1   no
 9+    9     0     1   no
 9-    9     0    -1   no
11+   11     0     1   no
11-   11     0    -1   no
-----------------------------------------------------------------------
```

The main difference from the *LS*-case is that target *c*-files are now provided for each *J*-level. In the MCHF package, the Breit–Pauli corrections are included in the framework of the CI calculations, and the solutions are recorded in the *j*-files, which contain the configuration expansions for all *J*-values. In order to obtain the *c*-file for a specific *J*-level, the user may use the utility **cfile** (see Section 13). For example, the command

cfile `even.j 1 4 2p2_3P2.c 0.0000001`

will generate the *c*-file for the state $2s^2 2p^2\ ^3P_2$, provided that CI results are recorded in the file `even.j`. The **bsr_par** file contains only one additional parameter, **mrel** $= 2$, indicating that the Hamiltonian now includes also all one-electron Breit–Pauli terms. The sequence of programs in the script file is the same as in the *LS* test run, with one correction for the BSR_BREIT program, namely:

bsr_breit
```
oper=1111000  klsp1=1  klsp2=12
```

The additional parameter **oper** indicates that the calculations of the angular coefficients also include the spin–orbit interaction. Recall that the angular coefficients are recorded in the databank files **int_bnk.nnn**, which can be used in repeated or corrected calculations.

14.5. Photoionization calculations

As an example of a photoionization calculation, we will consider the photodetachment of the negative carbon ion from its ground state $2p^3\ ^4S^o$. The final term can only be ^4P, and hence the **target** file only contains expansions for two partial waves, corresponding to the initial bound and the final continuum channels:

```
    hv + C-
---------------------------------------------------------------------
coupling = LS     !   LS coupling
nz = 6            !   nuclear charge
nelc = 6          !   number of electrons
---------------------------------------------------------------------
ntarg = 7         !   number of target states
---------------------------------------------------------------------
2p2_3P.c
2p2_1D.c
2p2_1S.c
2p3_5So.c
2p3s_3Po.c
2p3s_1Po.c
2p3_3Do.c
---------------------------------------------------------------------
nlsp =    2       !   number of partial waves
---------------------------------------------------------------------
4So   0     4    -1   p_4So.c
4Pe   1     4     1   no
---------------------------------------------------------------------
```

Note that the description of the initial state through the close-coupling expansion is optional, because the BSR complex also allows to represent the initial state as a real bound state obtained separately from independent MCHF calculations. A typical run of the BSR program for the case of photoionization calculations is described by the following script file:

```
bsr_prep

bsr_conf
bsr_breit   klsp1=1
bsr_mat     klsp1=1
bsr_hd      klsp1=1  itype=-1  msol=1

bsr_breit   klsp1=2
bsr_mat     klsp1=2

bsr_hd      klsp1=2  itype=1  idiag=1

mult        cfg.001 cfg.002 E1

bsr_dmat    cfg.001 cfg.002 b p

bsr_phot
```

We generate the bound state in the first partial wave, and the results are stored in the file **bound.001**. Then we generate the **h.002** and **rsol.002** files in the second partial wave, with the latter file containing all R-matrix solutions in the internal region. In comparison to the scattering calculations described in Section 14.1 above, we have two additional runs in order to generate the dipole matrix. The MULT program generates the databank for the dipole angular coefficients, and the BSR_DMAT program generates the dipole matrix file **d.002** itself. The **h.002** and **d.002** files are then used by the BSR_PHOT program for the photoionization calculations. The parameters and input energies are provided in the file **bsr_phot.inp** (see Section 11). The user can run the BSR_PHOT program several times for different energy intervals and with different energy steps. The results are accumulated in the files **photo.002** and **bsr_photo.002**. The file **photo.002** contains the total photoionization cross sections and can be used for a quick check of the calculations. The more detailed data recorded in the file **bsr_photo.002** may then be used for the calculation of channel cross sections or asymmetry parameters (see the description of the program utility **photo_tab** in Section 13). If we have several final continuum states with different total terms, the calculations are supposed to be carried out for each partial wave separately, and the total cross sections or asymmetry parameters may be generated again with the utility **photo_tab**. In order to verify that the system works correctly, the user should only compare the final photoionization cross sections in the file **photo.002** with the data provided in the file **photo.test**.

14.6. Bound-state calculations

This test is a simplified version of the calculation of oscillator strengths in carbon, presented in detail in [69]. The task was to obtain the oscillator strengths for transitions from the ground state to the Rydberg states $2s^22pns$ and $2s^22pnd$ for intermediate values of $n = 4$–8. In this range traditional methods, such as MCHF or the quantum defect method, are difficult to apply. The Rydberg series $2s^22pns$ and $2s^22pnd$ strongly interact with each other through both electrostatic and spin–orbit interaction. Besides, there are also strong corrections from core-valence and inner-core correlations. The test presents the simplest model of describing the $2s^22pns$ and $2s^22pn$ series though the close-coupling expansion with five target states: $2s^22p\ ^2P^o$, $2s2p^2\ ^2S$, ^2P, ^2D, and $2p^3\ ^2P^o$. The $2s2p^2$ states are included to simulate the core-valence correlation due to the strong 2s-2p transition, whereas the inclusion of the $2p^3$ state accounts, to some extent, for the inner-core correlation. The target file for this case has the following form:

```
   C - bound
------------------------------------------------------------------------
coupling = LS       !    LS coupling
nz = 6              !    nuclear charge
nelc = 5            !    number of electrons
------------------------------------------------------------------------
ntarg = 5           !    number of target states
------------------------------------------------------------------------
2s2_2p_2Po.c
2s_2p2_2D.c
2s_2p2_2P.c
2s_2p2_2S.c
2p3_2Po.c
------------------------------------------------------------------------
nlsp =      4       !    number of partial waves
------------------------------------------------------------------------
0-    0     0     -1    p0.c
1-    2     0     -1    p1.c
2-    4     0     -1    p2.c
3-    6     0     -1    p3.c
------------------------------------------------------------------------
```

All target states in this example are represented by simple single-configuration wavefunctions, but with term-dependent 2s and 2p one-electron orbitals. We are interested in the final J-level splitting, so the partial waves are constructed for total J-values from 0 to 3. Note that the target states here are represented in *LS* coupling. Hence this test is an example of using the *LSJ* coupling in the framework of the present BSR complex. The **bsr_par** input file for this test has the form:

```
# bsr_conf
<   2s|   ks>=0
<   2p|   kp>=0
max_LT = 7
max_ST = 3
max_ll = 3
#  bsr_mat:
mrel = 2
#  bsr_hd:
itype = -1
jmvc = 2
idiag = 2
Emax = -0.1
msol = 30
n_max= 12
qd_0 = 1.5
qd_1 = 0.6
qd_2 = 0.2
qd_3 = 0.1
```

The most important difference from the previous examples is the orthogonality condition imposed on both the 2s and 2p bound orbitals. It is due to the fact that in this case the same $2s2p^3$ bound states can be generated in different channels. This, in turn, may lead to an 'overloading' of the total overlap matrix as discussed in Section 5. The corresponding $2s2p^3$ states are included in the close-coupling expansion through the additional perturber expansion p0.c, p1.c, etc. For a more accurate representation of the $2s2p^3$ states, their wavefunctions were also obtained with a full re-optimization of the 2s and 2p orbitals. The **mrel** parameter in the above list indicates that the interaction matrix will include all one-electron Breit–Pauli terms, and **itype** $= -1$ means that it is a bound-state calculation with zero boundary condition. In order to obtain the states with high principal quantum numbers we use a border radius of 300 a.u. However, this does not lead to an increase in the number of B-splines, because we can use the exponential radial grid (see Section 12.2) for a bound-state calculation. The input parameters also contain the values for the expected quantum defects in the $2s^22pnl$ series. These parameters have no effect on the actual calculations; they are used only for a more appropriate labeling of the resulting states.

The run of the BSR programs in this test is described by the following script file:

```
bsr_prep
bsr_conf

bsr_breit   klsp1=1  klsp2=4  oper=1111000
bsr_mat     klsp1=1  klsp2=4
bsr_hd      klsp1=1  klsp2=4

bound_tab
mult        2s2_2p2_3P.c cfg.001 E1
mult        2s2_2p2_3P.c cfg.002 E1
mult        2s2_2p2_3P.c cfg.003 E1
mult        2s2_2p2_3P.c cfg.004 E1

bsr_dmat    2s2_2p2_3P0.c cfg.002 c b
bsr_dmat    2s2_2p2_3P1.c cfg.001 c b
bsr_dmat    2s2_2p2_3P1.c cfg.002 c b
bsr_dmat    2s2_2p2_3P1.c cfg.003 c b
bsr_dmat    2s2_2p2_3P2.c cfg.002 c b
bsr_dmat    2s2_2p2_3P2.c cfg.003 c b
bsr_dmat    2s2_2p2_3P2.c cfg.004 c b
```

After running the BSR_HD program we obtained the set of **bound.nnn** files with solutions for a given J-value. The final list of bound states in the **bound.nnn** files may be obtained with the program utility **bound_tab**. The initial states $2s^22p^2\ ^3P_{0,1,2}$ in this test are supposed to have been obtained in separate configuration-interaction calculations, with a list of configurations being placed in the file **2s2_2p2_3P.c** and the J-dependent expansion coefficients in the files **2s2_2p2_3Pj.c**. First, we run the MULT program to obtain all the required dipole angular coefficients in the databank **mult_bnk**. Then we run the program BSR_DMAT to calculate the oscillator strengths for each combination of initial and final J-values. Finally, the oscillator strengths from different runs are accumulated in the file **zf_res**. The file **zf_res.test** contains the correct results provided for comparison.

Acknowledgements

I am very grateful to Prof. Charlotte Froese Fischer, who proposed and initiated this work and also provided her B-spline library and suggested many useful improvements to the code. I am grateful to Dr. S. Tayal and Prof. T. Gorczyca for assisting with the first implementation of the code in scattering calculations and for helpful discussions at different stages of this work. I would like to thank Prof. K. Bartschat for many useful discussions, extensive testing of the code, and careful reading of the manuscript. This work was supported by the United States National Science Foundation under grant PHY-0311161.

References

[1] P.G. Burke, A. Hibbert, W.D. Robb, J. Phys. B 4 (1971) 153.
[2] P.G. Burke, W.D. Robb, Adv. At. Mol. Phys. 11 (1975) 143.
[3] P.G. Burke, K.A. Berrington, Atomic and Molecular Processes: An R-Matrix Approach, IOP Publishing, Bristol, 1993.
[4] K.A. Berrington, W.B. Eissner, P.H. Norrington, Comput. Phys. Comm. 92 (1995) 290.
[5] K.A. Berrington, P.G. Burke, J.J. Chang, A.T. Chivers, W.D. Robb, K.T. Taylor, Comput. Phys. Comm. 8 (1974) 149.
[6] K.A. Berrington, P.G. Burke, M. LeDourneuf, W.D. Robb, K.T. Taylor, Vo Ky Lan, Comput. Phys. Comm. 14 (1978) 367.
[7] N.S. Scott, K.T. Taylor, Comput. Phys. Comm. 25 (1982) 347.
[8] A.G. Sunderland, C.J. Noble, V.M. Burke, P.G. Burke, Comput. Phys. Comm. 143 (2002) 311.
[9] K.G. Dyall, I.P. Grant, C.T. Johnson, F.A. Parpia, E.P. Plummer, Comput. Phys. Comm. 55 (1989) 425.
[10] P.H. Norrington, I.P. Grant, J. Phys. B 20 (1987) 4869; see also http://web.am.qub.ac.uk/DARC/.
[11] K. Bartschat, Comput. Phys. Comm. 75 (1993) 219.
[12] P.G. Burke, M.P. Scott, Electron and Positron Scattering from Atoms and Ions, in: K. Bartschat (Ed.), Computational Atomic Physics, Springer, Heidelberg, 1996.
[13] P.G. Burke, C.J. Noble, M.P. Scott, Proc. Roy. Soc. A 410 (1987) 911.
[14] U. Fano, C.M. Lee, Phys. Rev. Lett. 31 (1973) 1573.
[15] M. Plummer, C.J. Noble, J. Phys. B 32 (1999) L345.
[16] P.J.A. Buttle, Phys. Rev. 160 (1967) 719.
[17] K. Bartschat, E.T. Hudson, M.P. Scott, P.G. Burke, V.M. Burke, J. Phys. B 29 (1996) 115.

[18] T.W. Gorczyca, N.R. Badnell, J. Phys. B 30 (1997) 3897.
[19] T.W. Gorczyca, F. Robicheaux, M.S. Pindzola, D.C. Grifin, N.R. Badnell, Phys. Rev. A 52 (1995) 3877.
[20] K. Bartschat, Comput. Phys. Comm. 114 (1998) 168.
[21] A. Hibbert, Comput. Phys. Comm. 1 (1970) 359.
[22] A. Hibbert, C. Froese Fischer, Comput. Phys. Comm. 64 (1991) 417.
[23] A. Hibbert, R. Glass, C. Froese Fischer, Comput. Phys. Comm. 64 (1991) 455.
[24] O. Zatsarinny, Comput. Phys. Comm. 98 (1996) 235.
[25] O. Zatsarinny, C. Froese Fischer, Comput. Phys. Comm. 124 (2000) 247.
[26] H. Bachau, E. Cormier, P. Decleva, J.E. Hansen, F. Martin, Rep. Prog. Phys. 64 (2001) 1815.
[27] H.W. van der Hart, J. Phys. B 30 (1997) 453.
[28] V.M. Burke, C.J. Noble, Comput. Phys. Comm. 85 (1995) 471.
[29] O. Zatsarinny, C. Froese Fischer, J. Phys. B 33 (2000) 313.
[30] O. Zatsarinny, S.S. Tayal, J. Phys. B 34 (2001) 3383.
[31] O. Zatsarinny, S.S. Tayal, J. Phys. B 35 (2002) 241.
[32] O. Zatsarinny, T.W. Gorczyca, C. Froese Fischer, J. Phys. B 35 (2002) 4161.
[33] O. Zatsarinny, T.W. Gorczyca, Abstracts of XIII ICPEAC, We026, Th006, 2003.
[34] O. Zatsarinny, K. Bartschat, J. Phys. B 37 (2004) 2173.
[35] O. Zatsarinny, K. Bartschat, J. Phys. B 37 (2004) 4693.
[36] M.A. Crees, M.J. Seaton, P.M.H. Wilson, Comput. Phys. Comm. 15 (1978) 23.
[37] R.J.W. Henry, S.P. Rountree, E.R. Smith, Comput. Phys. Comm. 23 (1981) 233.
[38] I. Bray, A.T. Stelbovics, Comput. Phys. Comm. 85 (1995) 1.
[39] C. Froese Fischer, M. Idrees, Comput. Phys. 3 (1989) 53.
[40] T. Brage, C. Froese Fischer, G. Miecznik, J. Phys. B 25 (1992) 5289.
[41] M. Venuti, P. Decleva, J. Phys. B 30 (1997) 4839.
[42] J. Xi, C. Froese Fischer, Phys. Rev. A 59 (1999) 307.
[43] M.A. Crees, Comput. Phys. Comm. 19 (1980) 103.
[44] M.J. Seaton, J. Phys. B 5 (1972) L91.
[45] H.E. Saraph, J. Phys. B 13 (1980) 3129.
[46] C. Mendoza, J. Phys. B 14 (1981) 397.
[47] M.J. Seaton, J. Phys. B 18 (1985) 2111.
[48] K.A. Berrington, M.J. Seaton, J. Phys. B 18 (1985) 2587.
[49] T. Brage, C. Froese Fischer, J. Phys. B 27 (1994) 5467.
[50] T. Brage, C. Froese Fischer, Phys. Scripta 49 (1994) 651.
[51] A.I. Akhiezer, V.B. Berestetsky, Quantum Electrodynamics, Interscience, New York, 1960.
[52] C. Froese Fischer, T. Brage, P. Jönsson, Computational Atomic Structure. An MCHF Approach, IOP Publishing, Bristol, 1997.
[53] A. Hibbert, Comput. Phys. Comm. 9 (1975) 141.
[54] W.B. Eissner, M. Jones, H. Nussbaumer, Comput. Phys. Comm. 8 (1974) 270.
[55] M. Bentley, J. Phys. B 27 (1994) 637.
[56] C. Froese Fischer, M.R. Godefroid, A. Hibbert, Comput. Phys. Comm. 64 (1991) 486.
[57] P.O. Löwdin, Phys. Rev. 97 (1955) 1474.
[58] D.A. Varshalovich, A.N. Moskalev, V.K. Khersonskij, Quantum Theory of Angular Momentum, World Scientific, Singapore, 1988.
[59] N. R Badnell, J. Phys. B 32 (1999) 5583; see also http://amdpp.phys.strath.ac.uk/UK_RmaX/codes.html.
[60] C. de Boor, A Practical Guide to Splines, Springer, New York, 1978.
[61] C. Froese Fischer, Int. J. Supercomp. Appl. 5 (1991) 5.
[62] C. Froese Fischer, in: T.N. Chang (Ed.), Many-Body Theory of Atomic Structure, Photoionization, World Science, Singapore, 1992.
[63] C. Froese Fischer, W. Guo, Z. Shen, Int. J. Quant. Chem. 42 (1992) 849.
[64] Y. Qiu, C. Froese Fischer, J. Comput. Phys. 156 (1999) 257.
[65] J. Sapirstein, W.R. Johnson, J. Phys. B 29 (1996) 5213.
[66] J.E. Hansen, M. Bently, H.W. van der Hart, M. Landtman, G.M.S. Lister, Y.-T. Shen, N. Vaeck, Phys. Scripta T 47 (1993) 7.
[67] T.N. Chang, T.K. Fang, Phys. Rev. A 52 (1995) 2638.
[68] C. Froese Fischer, Comput. Phys. Comm. 64 (1991) 369.
[69] O. Zatsarinny, C. Froese Fischer, J. Phys. B 35 (2002) 4669.
[70] O. Zatsarinny, K. Bartschat, L. Bandurina, V. Gedeon, Phys. Rev. A 71 (2005) 042702.