MOLSCAT, BOUND and FIELD

Version 2020.0

User Manual

Jeremy M. Hutson and C. Ruth Le Sueur

February 14, 2020

Contents

Chapter 1

Preamble

1.1 Citing the programs

Any publication that uses MOLSCAT, BOUND or FIELD should cite both the version of the program used:

Jeremy M. Hutson and C. Ruth Le Sueur, MOLSCAT: a program for non-reactive quantum scattering calculation on atomic and molecular collisions, Version 2020.0, https://github.com/molscat/molscat.

Jeremy M. Hutson and C. Ruth Le Sueur, BOUND: a program for bound states of interacting pairs of atoms and molecules, Version 2020.0, https://github.com/molscat/molscat.

Jeremy M. Hutson and C. Ruth Le Sueur, FIELD: a program for bound states of interacting pairs of atoms and molecules as a function of external field, Version 2020.0, https://github.com/molscat/molscat.

and the published paper(s):

Jeremy M. Hutson and C. Ruth Le Sueur, 'MOLSCAT: a program for non-reactive quantum scattering calculations on atomic and molecular collisions',
Computer Physics Communications 241, 9-18 (2019):
https://doi.org/10.1016/j.cpc.2019.02.014.

Jeremy M. Hutson and C. Ruth Le Sueur, 'BOUND and FIELD: programs for calculating bound states of interacting pairs of atoms and molecules",
Computer Physics Communications 241, 1-8 (2019):
https://doi.org/10.1016/j.cpc.2019.02.017.

Pre-publication versions of the papers are also available from arXiv:
https://arxiv.org/abs/1811.09584
https://arxiv.org/abs/1811.09111
and the current and previous versions of this documentation are available from
https://arxiv.org/abs/1903.06755.

1.2 Licensing

These programs are free software: you can redistribute them and/or modify them under the terms of the GNU General Public License, version 3, as published by the Free Software Foundation. The full text of the license is available from https://www.gnu.org/licenses/ and is included in the file `COPYING` included in the distribution.

1.3 Updates

Program updates are made available via github at `https://github.com/molscat/molscat`. No github user account is needed. If you are using a Linux machine with git installed, the commands to create a directory containing the program source code and associated files are

```
mkdir molscat
```

```
git clone https://github.com/molscat/molscat molscat
```

On subsequent occasions you can update your source code to the latest distributed version simply by navigating to the molscat directory and issuing the command

```
git pull
```

1.4 Scope of the programs

MOLSCAT is a general-purpose package for performing non-reactive quantum scattering calculations for atomic and molecular collisions using coupled-channel methods. Simple atom-molecule and molecule-molecule collision types are coded internally and additional ones may be handled with plug-in routines. Plug-in routines may include external magnetic, electric or photon fields (and combinations of them). Simple interaction potentials are coded internally and more complicated ones may be handled with plug-in routines.

BOUND is a general-purpose package for performing calculations of bound-state energies in weakly bound atomic and molecular systems using coupled-channel methods. It solves the same sets of coupled equations as MOLSCAT, and can use the same plug-in routines if desired, but with different boundary conditions.

FIELD is a development of BOUND that locates external fields at which a bound state exists with a specified energy. One important use is to locate the positions of magnetically tunable Feshbach resonance positions in ultracold collisions.

Versions of these programs before 2019.0 were released separately. However, there is a significant degree of overlap between their internal structures and usage specifications. This manual therefore describes all three, with careful identification of parts that are specific to one or two of the programs.

The authors would be grateful to know of any bugs encountered in any of the programs or errors in this documentation.

1.5 Program capabilities

MOLSCAT, BOUND and FIELD all construct sets of coupled differential equations that represent the Hamiltonian for atom-atom, atom-molecule or molecule-molecule interactions. They have built-in capabilities to generate the coupled equations for

- Atom + linear rigid rotor;
- Atom + vibrating diatom;
- Linear rigid rotor + linear rigid rotor;
- Atom + symmetric top;
- Atom + asymmetric top;
- Asymmetric top + linear molecule;
- Atom + rigid corrugated surface: diffractive (elastic) scattering and band structure.

The programs can set up the coupled equations for

- Full close-coupling calculations, with no dynamical approximations;
- Effective potential approximation;
- CS (coupled-states/centrifugal-sudden) approximation;
- Decoupled L-dominant approximation;
- Infinite-order sudden approximation (MOLSCAT only).

The programs also include an interface for a plug-in basis-set suite to set up other sets of coupled equations. Basis-set suites can be programmed to take account of one or more external fields, such as electric, magnetic and photon fields. Two such suites are included in this distribution, for

- ^{1}S atom + $^3\Sigma$ diatom in a magnetic field;
- Alkali-metal atom + alkali-metal atom in a magnetic field, including hyperfine structure.

Input energies may be specified with respect to a particular scattering threshold if desired. For basis sets that take account of external field(s), the reference energy is often field-dependent.

The interaction potential between the interacting species may be supplied in a variety of ways. Very simple potentials may be supplied as explicit input data. More sophisticated potentials may be supplied as external routines that provide either:

- the coefficients for an expansion of the potential in suitable internal coordinates at a supplied value of the interspecies distance; or
- the values of the potential for a supplied set of internal coordinates and the interspecies distance; the programs then integrate over the internal coefficients to obtain the expansion coefficients.

The programs loop over good quantum numbers such as the total angular momentum and parity of the interacting pair, and construct a separate set of coupled equations for each set of values of the good quantum numbers.

The programs solve the coupled equations using a variety of numerical methods, all of which propagate the wavefunction matrix or its log-derivative across a range of values of the interspecies separation.

MOLSCAT performs scattering calculations at a list or grid of energies where there is at least one

channel that is asymptotically open (energetically accessible at long range). It propagates the solutions outwards from short range to a point at long range where the interaction potential is insignificant. It matches the solution at long range to analytic functions that describe the solutions in the absence of the interaction potential and obtains the scattering S matrix. It then has options to

- Combine S matrices from different values of total angular momentum and other symmetries to calculate state-to-state integral cross sections;
- Combine S matrices from different values of total angular momentum and parity to calculate spectroscopic line-shape cross sections;
- Output S matrices to files for subsequent processing to calculate differential cross sections or generalised transport, relaxation and Senftleben-Beenakker cross sections;
- Converge on and characterise scattering resonances or predissociating / quasibound states of the collision complex through their signature in the eigenphase sum (multichannel phase shift);
- Output K matrices to files for subsequent processing to characterise scattering resonances;
- Calculate low-energy scattering properties such as scattering lengths, scattering volumes and effective ranges;
- Converge on and characterise low-energy Feshbach resonances in the scattering length as a function of external field(s).

BOUND seeks bound states in a specified range of energies, usually at energies where all channels are asymptotically closed (energetically inaccessible at long range). It propagates the solutions outwards from short range and inwards from long range to a matching point in the classically allowed region. It compares the outwards and inwards solutions and attempts to converge on energies where the wavefunction is both continuous and continuously differentiable at the matching point. These are the bound-state energies of the system.

BOUND has options to

- Calculate expectation values without explicit wavefunctions;
- Calculate bound-state wavefunctions and output them to files for subsequent processing.

For basis sets that take account of external field(s), BOUND can loop over a list or grid of values of the external fields and locate bound states at each field.

FIELD operates in a similar way to BOUND but seeks values of the external field(s) at which bound states exist at a specified list or grid of energies.

1.6 Program limitations

The programs propagate coupled equations with respect to a single integration variable R (usually the interparticle distance). They require that the kinetic energy operator in the variable R can be written in the form $[f(R)]^{-1}(d^2/dR^2)f(R)$, where $f(R)$ is usually a power R^n.

The programs can handle interactions of two structured particles, such as complex atoms or molecules, but they are not generally applicable to problems involving more than 2 particles,

unless they can be expressed as a set of coupled equations in a single variable.

The programs are efficient only if the wavefunction at all physically significant values of R can be expanded compactly in terms of an R-independent basis set of functions in the remaining variables. This is often not true for strongly bound states of polyatomic molecules, and different methods are preferable in such cases.

The programs cannot handle interaction potentials that are non-local in the integration variable, as sometimes occur in nuclear scattering problems.

MOLSCAT cannot apply asymptotic boundary conditions in two different integration variables, as is common in reactive scattering calculations.

MOLSCAT currently applies boundary conditions that are appropriate only if the interaction potential decays faster than R^{-2} at long range. Modifications would be needed to handle scattering boundary conditions for Coulomb potentials.

1.7 Structure of documentation

In this manual, coloured vertical bars denote sections that apply to a subset of the programs, as follows:

- MOLSCAT
- BOUND
- FIELD
- BOUND and FIELD
- MOLSCAT and FIELD
- MOLSCAT and BOUND

This documentation is structured as follows:

- Chapter 2 provides a brief description of the theory behind the programs.
- Chapter 3 provides an introduction to use of the programs, with some basic examples.
- Chapter 4 describes the arrays used to specify basis sets and the input data required for the built-in interaction types.
- Chapter 5 describes how to specify interaction potentials, including the specification for plug-in potential routines `VINIT`/`VSTAR` and `VRTP`.
- Chapter 6 describes how to specify energies in MOLSCAT and FIELD and energy ranges for bound states in BOUND and FIELD. It also describes how to select a reference threshold to be used as a (potentially field-dependent) zero of energy.
- Chapter 7 describes how to specify external fields in MOLSCAT and BOUND and field ranges for bound states in FIELD.
- Chapter 8 describes how to control the propagators used to solve the coupled equations.
- Chapter 9 describes how to specify which properties are calculated from S matrices by MOLSCAT.
- Chapter 10 describes how to control bound-state calculations in BOUND and FIELD.
- Chapter 11 provides a complete list of the input variables and arrays, in the form of alphabetically organised quick-reference lists.
- Chapter 12 describes how to control the level of printed output.
- Chapter 13 describes the auxiliary and scratch files that may be produced or used.
- Chapter 14 describes the example input files provided with the programs to illustrate their capabilities, the output they each produce, and how to interpret it.

Up to this point, the user needs only limited knowledge of internal variables and subroutines (except for any plug-in routines used for calculating the interaction potential). The remainder of the documentation is intended for users who wish to install the program on a new computer system, program more complicated interaction potentials, implement new plug-in basis-set routines, use the plug-in basis-set routines provided, or modify other parts of the program.

- Chapter 15 describes the files provided with the distribution and how to build executables. It includes information about potentially machine-dependent features and describes program features likely to be needed by users who wish to program their own plug-in interaction potential routines or plug-in basis-set suites.
- Chapter 16 describes the (now seldom used) option to supply a complete `POTENL` routine to evaluate the set of potential expansion coefficients, rather than the simpler `VINIT`/`VSTAR` or `VRTP` routines described in chapter 5.
- Chapter 17 sets out the specification of a plug-in basis-set suite of routines which describe a basis set and calculate coupling coefficients different from the built-in interaction types. It is intended principally for users who wish to program their own suite.
- Chapter 18 describes the two plug-in basis-set suites provided as part of the distributed code.
- Chapter 19 describes the plug-in potential routines provided as part of the distributed code.

1.8 Program history

The MOLSCAT program was originally written by Sheldon Green in the 1970s, incorporating propagators from several different authors and adapting them to share the same input / output structures and mechanisms for generating matrix elements of the molecular Hamiltonian and interaction potential. Early versions of the program handled atom-molecule and molecule-molecule scattering, with a variety of additional coupling cases added between versions 1 (1973) and 7 (1979) [1]. Both full space-fixed close-coupling calculations and decoupling approximations such as coupled states / centrifugal-sudden (CS), infinite-order sudden (IOS) and decoupled L-dominant (DLD) approximations were implemented. The program calculated integral elastic, state-to-state inelastic and line-shape cross sections internally and wrote S matrices to a file that could be used for post-processors, including DCS [2] for differential cross sections and SBE [3] for cross sections associated with transport and relaxation properties and Senftleben-Beenakker effects.

Jeremy Hutson became a co-author of MOLSCAT in 1982 and collaborated with Sheldon Green on its development until Green's death in 1995. Major changes in this period included new coupling cases, including the diffractive scattering of atoms from crystal surfaces, the replacement of some older propagators with new ones, and code to handle scattering resonances (with post-processor RESFIT [4] for fitting resonance parameters). In addition, an interface was added in version 11 (1992) to allow the inclusion of plug-in coupling cases with angular

momentum algebra and/or interaction potential expansions that were not already present in the code.

The last version of MOLSCAT produced by Green and Hutson in collaboration was version 14 in 1994. This was distributed via the CCP6 collaboration in the UK [5] and via the NASA GISS website. Version 14 formed the basis of a parallel version named PMP MOLSCAT, produced by George McBane [6].

BOUND was originally written by Jeremy Hutson in 1984 to calculate bound states of Van der Waals complexes by coupled-channel methods, using the same structures as MOLSCAT to generate the coupled equations. Subsequent versions incorporated basis-set enhancements as they were made in MOLSCAT. A fundamental change was made in version 5 (1993) to base the convergence algorithm on individual eigenvalues of the log-derivative matching matrix [7], rather than its determinant. Versions 4 (1992) and 5 (1993) [8] were distributed via CCP6.

MOLSCAT and BOUND were extended to handle calculations in external electric and magnetic fields in 2007. FIELD was written by Jeremy Hutson in 2010, using the same structures as MOLSCAT and BOUND to generate the coupled equations but designed to locate bound states as a function of external field at fixed energy, rather than as a function of energy.

There was no fully documented publication of MOLSCAT or BOUND between 1994 and 2019. FIELD was not formally published until 2019. However, there were continuing enhancements to the capabilities in the intervening years, and updates were provided privately to selected users. In this documentation we treat all changes from MOLSCAT version 14 and BOUND version 5 as new in version 2019.0, since that was the first time they were collected together and documented.

1.9 Principal changes in version 2019.0

The basis-set plug-in mechanism was extended to allow propagation in basis sets that are not eigenfuctions of the internal Hamiltonian $H_{\rm intl}$. This made implementing new types of system much simpler than before, especially where the individual interaction partners have complicated Hamiltonians.

This functionality was used to add new capabilities to carry out calculations in external fields (electric, magnetic, and/or photon) and to loop over (sets of) values of the fields.

Where necessary, MOLSCAT now tranformed the propagated wavefunction or log-derivative matrix to a basis set diagonal in $H_{\rm intl}$ before matching to long-range functions to extract the S matrix.

MOLSCAT was extended to process the S matrix to calculate scattering lengths (or volumes or hypervolumes) a_L [actually k-dependent complex scattering lengths/volumes $a_L(k)$] for any low-energy scattering channels.

MOLSCAT was extended to extrapolate the real part of $a_0(k)$ to $k = 0$ and calculate the effective range $r_{\rm eff}$.

MOLSCAT was extended to converge on and characterise Feshbach resonances as a function of external field. This is implemented for both the elastic case, where resonances appear as

poles in the scattering length/volume, and the inelastic case, where resonances are decayed and have more complicated signatures.

MOLSCAT was extended to calculate a multichannel scattering wavefunction for flux incoming in a single channel and outgoing in all open channels.

BOUND and FIELD separated the functions of $R_{\rm mid}$, the distance where the calculation switches between short-range and long-range propagators, and $R_{\rm match}$, the distance where the incoming and outgoing wavefunctions are matched.

BOUND and FIELD were modified to calculate the node count from an outwards propagation from $R_{\rm min}$ to $R_{\rm match}$, an inwards propagation from $R_{\rm max}$ to $R_{\rm match}$, and the number of negative eigenvalues of the log-derivative matching matrix. This eliminated the need for a third propagation from $R_{\rm match}$ to $R_{\rm min}$ or $R_{\rm max}$.

All three programs implemented a more general mechanism for combining propagators for use at short and long range, which allows any sensible combination.

All the programs allowed more general choices of log-derivative boundary conditions at the starting points for propagation.

A new propagation approach [9] was included, implemented by George McBane. This takes advantage of the symplectic nature of the multi-channel radial Schrödinger equation and reformulates it so that symplectic integrators (SIs) may be used to propagate solutions of the coupled equations. Coefficients for two SIs were included: the five-step fourth-order method of Calvo and Sanz-Serna (CS4) [10] and the six-step fifth-order method of McLachlan and Atela (MA5) [11]. The approach was coded so that other SIs could easily be implemented if desired.

1.10 Principal changes in version 2019.1

The fundamental physical constants used by default were updated to the 2018 CODATA recommended values.

The terminology used in the output from BOUND and FIELD was modified to introduce a *state number* that is equal to the node count immediately above the state concerned.

The output on the S-matrix save file was modified to allow more general indexing of channel energies. This is an incompatible change for `ITYP` = 3, 4, 5, 6, so the `ISAVEU` format version number `IPROGM` (section 13.5.1) was increased to 19.

Minor bug fixes as documented on github.

1.11 Principal changes in version 2020.0

MOLSCAT can now converge on and characterise a scattering resonance or quasibound state as a function of energy, as described in ref. [12]; see section 9.9.

MOLSCAT now converges on resonances in the scattering length using the procedures of ref.

[13] but with the improved algorithm for point selection developed in ref. [12]; this can provide widths that are stabler with respect to variations in the Hamiltonian.

MOLSCAT now evaluates integral cross sections for collisions of identical molecules using a definition that gives the same result with and without identical-particle symmetry when the two molecules are in the same initial or final state; see section 4.5.3.

BOUND and FIELD now allow the log-derivative matching point R_{match} to be at the end of the propagation range (at R_{min} or R_{max}) as well as part-way along it.

BOUND can now calculate the expectation value of the operator for an external field and use it to evaluate the derivative of the bound-state energy with respect to field, for both the absolute energy and the energy relative to threshold.

The potential routine `vstar-Tiemann.f` and its associated data modules, which implement interaction potentials for alkali-metal pairs that use the functional form of Tiemann and coworkers (section 19.5), have been generalised to allow the use of potential parameters that do not give exact continuity at the matching points. This is not generally recommended, but it allows the use of potential parameters exactly as published for comparison with other routines.

A new structure has been introduced to allow reinitialisation of the potential routine for each set of external fields. The default behaviour is unchanged, but the reinitialisation may be used to interpret a scaling factor `SCALAM` (treated as a field) as a chosen linear or non-linear potential parameter. This allows use of the built-in convergence capabilities of MOLSCAT or FIELD to adjust the parameter to reproduce precisely a required value of an observable quantity such as a resonance position or bound-state energy.

The names and structures of the dependency lists in the supplied makefile have been changed (section 15.13) to simplify the substitution of some routines by special adapted versions (expert use only).

Minor bug fixes as documented on github.

Chapter 2

Theory

2.1 The Hamiltonian

There are many problems in quantum mechanics in which the total Hamiltonian of the system may be written

$$H = -\frac{\hbar^2}{2\mu} R^{-1} \frac{d^2}{dR^2} R + \frac{\hbar^2 \hat{L}^2}{2\mu R^2} + H_{\text{intl}}(\xi_{\text{intl}}) + V(R, \xi_{\text{intl}}), \tag{2.1}$$

where R is a radial coordinate describing the separation of two particles and ξ_{intl} represents all the other coordinates in the system. H_{intl} represents the sum of the internal Hamiltonians of the isolated particles, and depends on ξ_{intl} but not R, and $V(R, \xi_{\text{intl}})$ is an interaction potential. The operator $\hbar^2 \hat{L}^2 / 2\mu R^2$ is the centrifugal term that describes the end-over-end rotational energy of the interacting pair.

The internal Hamiltonian H_{intl} is a sum of terms for the two particles 1 and 2,

$$H_{\text{intl}}(\xi_{\text{intl}}) = H^{(1)}_{\text{intl}}(\xi^{(1)}_{\text{intl}}) + H^{(2)}_{\text{intl}}(\xi^{(2)}_{\text{intl}}), \tag{2.2}$$

with eigenvalues $E_{\text{intl},i} = E^{(1)}_{\text{intl},i} + E^{(2)}_{\text{intl},i}$, where $E^{(1)}_{\text{intl},i}$ and $E^{(2)}_{\text{intl},i}$ are energies of the separated monomers 1 and 2. The individual terms can vary enormously in complexity: each one may represent a structureless atom, requiring no internal Hamiltonian at all, a vibrating and/or rotating molecule, or a particle with electron and/or nuclear spins. The problems that arise in ultracold physics frequently involve pairs of atoms or molecules with electron and nuclear spins, often in the presence of external electric, magnetic or photon fields. All these complications can be taken into account in the structure of H_{intl} and the interaction potential $V(R, \xi_{\text{intl}})$, which may both involve terms dependent on spins and external fields.

2.2 Coupled-channel approach

A standard computational approach for solving the Schrödinger equation for the Hamiltonian (2.1) is the *coupled-channel* approach, which handles the radial coordinate R by direct

numerical propagation on a grid, and all the other coordinates using a basis set. In the coupled-channel approach, the total wavefunction is expanded

$$\Psi(R,\xi_{\rm intl}) = R^{-1}\sum_j \Phi_j(\xi_{\rm intl})\psi_j(R), \tag{2.3}$$

where the functions $\Phi_j(\xi_{\rm intl})$ form a complete orthonormal basis set for motion in the coordinates $\xi_{\rm intl}$ and the factor R^{-1} serves to simplify the form of the radial kinetic energy operator. The wavefunction in each *channel* j is described by a radial *channel function* $\psi_j(R)$. The expansion (2.3) is substituted into the total Schrödinger equation, and the result is projected onto a basis function $\Phi_i(\xi_{\rm intl})$. The resulting coupled differential equations for the channel functions $\psi_i(R)$ are

$$\frac{d^2\psi_i}{dR^2} = \sum_j \left[W_{ij}(R) - \mathcal{E}\delta_{ij}\right]\psi_j(R), \tag{2.4}$$

where δ_{ij} is the Kronecker delta, $\mathcal{E} = 2\mu E/\hbar^2$, E is the total energy, and

$$W_{ij}(R) = \frac{2\mu}{\hbar^2}\int \Phi_i^*(\xi_{\rm intl})[\hbar^2\hat{L}^2/2\mu R^2 + H_{\rm intl} + V(R,\xi_{\rm intl})]\Phi_j(\xi_{\rm intl})\,d\xi_{\rm intl}. \tag{2.5}$$

The different equations are coupled by the off-diagonal terms $W_{ij}(R)$ with $i \neq j$.

The coupled equations may be expressed in matrix notation,

$$\frac{d^2\boldsymbol{\psi}}{dR^2} = \left[\mathbf{W}(R) - \mathcal{E}\mathbf{I}\right]\boldsymbol{\psi}(R). \tag{2.6}$$

If there are N basis functions included in the expansion (2.3), $\boldsymbol{\psi}(R)$ is a column vector of order N with elements $\psi_j(R)$, $\mathbf{I}$ is the $N \times N$ unit matrix, and $\mathbf{W}(R)$ is an $N \times N$ interaction matrix with elements $W_{ij}(R)$.

In general there are N linearly independent solution vectors $\boldsymbol{\psi}(R)$ that satisfy the Schrödinger equation subject to the boundary condition that $\boldsymbol{\psi}(R) \to 0$ in the classically forbidden region at short range. These N column vectors form a wavefunction matrix $\boldsymbol{\Psi}(R)$. The various propagators in MOLSCAT, BOUND and FIELD work either by propagating $\boldsymbol{\Psi}(R)$ and its radial derivative $\boldsymbol{\Psi}'(R)$ or by propagating the log-derivative matrix $\mathbf{Y}(R) = \boldsymbol{\Psi}'(R)[\boldsymbol{\Psi}(R)]^{-1}$.

The particular choice of the basis functions $\Phi_j(\xi_{\rm intl})$ and the resulting form of the interaction matrix elements $W_{ij}(R)$ depend on the physical problem being considered. The complete set of coupled equations often factorises into blocks determined by the symmetry of the system. In the absence of external fields, the *total angular momentum* $J_{\rm tot}$ and the *total parity* are conserved quantities. Different or additional symmetries arise in different physical situations. The programs are designed to loop over total angular momentum and parity, constructing a separate set of coupled equations for each combination and solving them by propagation. These loops may be repurposed for other symmetries when appropriate.

The programs can also handle interactions that occur in external fields, where the total angular momentum is no longer a good quantum number.

2.3 Convention for quantum numbers

In bound-state and scattering calculations, it is often necessary to distinguish between quantum numbers for the individual monomers and for the pair (supermolecule or interaction complex). It is a widely used convention to use lower-case letters for the individual monomers and upper-case letters for the pair. Because of this, monomer quantum numbers that are conventionally upper-case in single-molecule spectroscopy (J, K, M_J, F, etc.) are often converted to lower-case here (j, k, m_j, f, etc.), with the upper-case letters reserved for the corresponding quantum numbers of the interacting pair. Where monomer quantum numbers are needed for both species, they are indicated by a subscript 1 or 2 (j_1, j_2, etc.).

In keeping with this convention, the end-over-end angular momentum of the interacting pair is denoted L, rather than l.

2.4 Matrix of the interaction potential

In order to streamline the calculation of matrix elements for the propagation, MOLSCAT, FIELD and BOUND express the interaction potential in an expansion over the internal coordinates,

$$V(R, \xi_{\mathrm{intl}}) = \sum_{\Lambda} v_{\Lambda}(R) \mathcal{V}^{\Lambda}(\xi_{\mathrm{intl}}). \tag{2.7}$$

The specific form of the expansion depends on the nature of the interacting particles. The radial potential coefficients $v_\Lambda(R)$ may either be supplied explicitly, or generated internally by numerically integrating over ξ_{intl}. The R-independent coupling matrices $\boldsymbol{\mathcal{V}}^{\Lambda}$ with elements $\mathcal{V}^{\Lambda}_{ij} = \langle \Phi_i | \mathcal{V}^{\Lambda} | \Phi_j \rangle_{\mathrm{intl}}$ are calculated once and stored for use in evaluating $W_{ij}(R)$ throughout the course of a propagation.

2.5 Matrices of the internal and centrifugal Hamiltonians

Coupled-channel scattering theory is most commonly formulated in a basis set where $\hat{L}^2$ and H_{intl} are both diagonal. All the built-in coupling cases use basis sets of this type. The matrix of H_{intl} is $\langle \Phi_i | H_{\mathrm{intl}} | \Phi_j \rangle_{\mathrm{intl}} = E_{\mathrm{intl},i} \delta_{ij}$. The diagonal matrix elements of $\hat{L}^2$ are often of the form $L_i(L_i + 1)$, where the integer quantum number L_i (sometimes called the partial-wave quantum number) represents the end-over-end angular momentum of the two particles about one another.

However, the programs also allow the use of basis sets where one or both of $\hat{L}^2$ and H_{intl} are non-diagonal. If H_{intl} is non-diagonal, it is expanded as a sum of terms

$$H_{\mathrm{intl}}(\xi_{\mathrm{intl}}) = \sum_{\Omega} h_{\Omega} \mathcal{H}^{\Omega}_{\mathrm{intl}}(\xi_{\mathrm{intl}}), \tag{2.8}$$

where the h_Ω are scalar quantities, some of which may represent external fields if desired. The programs generate additional coupling matrices $\boldsymbol{\mathcal{H}}^{\Omega}$ with elements $\mathcal{H}^{\Omega}_{ij} = \langle \Phi_i | \mathcal{H}^{\Omega}_{\mathrm{intl}} | \Phi_j \rangle_{\mathrm{intl}}$.

These are also calculated once and stored for use in evaluating $W_{ij}(R)$ throughout the course of a propagation. A similar mechanism is used for basis sets where $\hat{L}^2$ is non-diagonal, with

$$\hat{L}^2 = \sum_{\Upsilon} \mathcal{L}^{\Upsilon}. \tag{2.9}$$

If H_{intl} is non-diagonal, the allowed energies $E_{\text{intl},i}$ of the pair of monomers at infinite separation are the eigenvalues of H_{intl}. The wavefunctions of the separated pair are represented by simultaneous eigenvectors of H_{intl} and $\hat{L}^2$.

2.6 Results of scattering calculations

The outcome of a collision process is usually described in quantum mechanics by the scattering matrix (S matrix), which contains information on the probability amplitudes and phases for the various possible outcomes. In simple cases (diagonal H_{intl} and $\hat{L}^2$), each possible outcome corresponds to one of the channels in the coupled equations. Alternatively, if H_{intl} is non-diagonal, each outcome corresponds to an *asymptotic channel* represented by one of the simultaneous eigenvectors of H_{intl} and $\hat{L}^2$ with energy eigenvalue $E_{\text{intl},i}$. Each asymptotic channel i is *open* if it is energetically accessible as $R \to \infty$ ($E_{\text{intl},i} \leq E$) or *closed* if it is energetically forbidden ($E_{\text{intl},i} > E$).

For each J_{tot} and symmetry block, solutions to the coupled equations are propagated from deep inside the classically forbidden region at short range to a distance at long range beyond which the interaction potential may be neglected. The wavefunction matrix $\mathbf{\Psi}(R)$ and its radial derivative (or the log-derivative matrix $\mathbf{Y}(R)$) are then matched to the analytic functions that describe the solutions of the Schrödinger equation in the absence of an interaction potential,

$$\mathbf{\Psi}(R) = \mathbf{J}(R) + \mathbf{N}(R)\mathbf{K}. \tag{2.10}$$

where the matrices $\mathbf{J}(R)$ and $\mathbf{N}(R)$ are diagonal and are made up of Ricatti-Bessel functions for the open channels and modified spherical Bessel functions for the closed channels.[1]

For each channel i, the Bessel function is of order L_i and its argument is $k_i R$, where k_i is the asymptotic wavevector such that $\hbar^2 k_i^2 / 2\mu = |E - E_{\text{intl},i}|$.

The real symmetric $N \times N$ matrix $\mathbf{K}$ is then converted to the S matrix,

$$\mathbf{S} = (\mathbf{I} + i\mathbf{K}_{\text{oo}})^{-1}(\mathbf{I} - i\mathbf{K}_{\text{oo}}), \tag{2.11}$$

where $\mathbf{K}_{\text{oo}}$ is the open-open portion of $\mathbf{K}$. $\mathbf{S}$ is a complex symmetric unitary matrix of dimension $N_{\text{open}} \times N_{\text{open}}$, where N_{open} is the number of open channels.

If $\hat{L}^2$ and H_{intl} are both diagonal, the asymptotic channels used for matching to Bessel functions are the same as the channels used to propagate the wavefunction matrix $\mathbf{\Psi}(R)$ or its log-derivative $\mathbf{Y}(R)$.

If $\hat{L}^2$ and/or H_{intl} is non-diagonal, MOLSCAT transforms $\mathbf{\Psi}(R)$ or $\mathbf{Y}(R)$ at $R = R_{\text{max}}$ into a basis set that diagonalises $\hat{L}^2$ and H_{intl}.

[1] These boundary conditions are appropriate only if the interaction potential decays faster than R^{-2} at long range. Different boundary conditions would be needed to handle long-range Coulomb potentials.

- If $\hat{L}^2$ is diagonal but $H_{\rm intl}$ is not, MOLSCAT constructs the matrix of $H_{\rm intl}$ for each value of L in turn, and diagonalises it. It uses the resulting eigenvectors to transform the corresponding block of $\mathbf{\Psi}(R_{\rm max})$ or $\mathbf{Y}(R_{\rm max})$ into the asymptotic basis set.
- If $\hat{L}^2$ is non-diagonal (whether $H_{\rm intl}$ is diagonal or not), MOLSCAT constructs the complete matrix of $H_{\rm intl}$ and diagonalises it. If there are degenerate eigenvalues $E_{{\rm intl},i}$ of $H_{\rm intl}$, it then constructs the matrix of $\hat{L}^2$ for each degenerate subspace and diagonalises it to obtain eigenvalues L_i and simultaneous eigenvectors of $\hat{L}^2$ and $H_{\rm intl}$. It uses the simultaneous eigenvectors to transform $\mathbf{\Psi}(R_{\rm max})$ or $\mathbf{Y}(R_{\rm max})$ into the asymptotic basis set.

Finally MOLSCAT uses the transformed $\mathbf{\Psi}(R_{\rm max})$ or $\mathbf{Y}(R_{\rm max})$, together with the eigenvalues $E_{{\rm intl},i}$ and L_i, to extract $\mathbf{K}$.

Experimental observables that describe completed collisions, such as differential and integral cross sections, and scattering lengths, can be written in terms of S-matrix elements. Cross sections typically involve a *partial-wave sum*,[2] with contributions from many values of $J_{\rm tot}$, except at the lowest kinetic energies (in the ultracold regime). By default MOLSCAT uses its S matrices to accumulate degeneracy-averaged state-to-state integral cross sections, which may be written

$$\sigma_{n_{\rm i}\to n_{\rm f}} = \frac{\pi}{g_{n_{\rm i}} k_{n_{\rm i}}^2} \sum_{\substack{J_{\rm tot}\\ M}} (2J_{\rm tot}+1) \sum_{\substack{i\in n_{\rm i}\\ f\in n_{\rm f}}} \left| \delta_{if} - S_{if}^{J_{\rm tot},M} \right|^2 . \tag{2.12}$$

Here $n_{\rm i}$ and $n_{\rm f}$ label initial and final levels (not states) of the colliding pair,[3] while i and f indicate the open channels arising from those levels for total angular momentum $J_{\rm tot}$ and symmetry block M. $g_{n_{\rm i}}$ is the degeneracy of level $n_{\rm i}$. The S matrices may optionally be used to compute line-shape cross sections (section 9.7), or be saved to a file for post-processing in order to obtain other kinds of collision properties.

MOLSCAT can calculate line-shape cross sections for the broadening, shifting and mixing of spectroscopic lines for most of the built-in coupling cases. Line-shape cross sections require scattering calculations for the upper and lower states of the spectroscopic transition at the same *kinetic* (not total) energy; if desired, the input energies are interpreted as kinetic energies and the total energies required are generated internally.

MOLSCAT has features to locate scattering resonances, which may produce sharp features in the energy-dependence of cross sections and may also be interpreted as predissociating states of Van der Waals complexes. It can calculate the S-matrix eigenphase sum, which is a generalisation of the scattering phase shift to multichannel problems. It can converge on resonances in the eigenphase sum and obtain their positions and widths, and can also calculate partial widths to individual open channels.

MOLSCAT can output S matrices to auxiliary files for later processing. Separate programs are available:

[2] Note that there is some inconsistency in the literature in the use of "partial wave" for multichannel scattering. It sometimes refers to the total angular momentum, and sometimes to the end-over-end angular momentum L of the colliding pair.

[3] For some interaction types, $n_{\rm i}$ and $n_{\rm f}$ each represent several quantum numbers, not just one.

- program `DCS` [2] to calculate differential cross sections;
- program `SBE` [3] to calculate generalised transport, relaxation and Senftleben-Beenakker cross sections;
- program `RESFIT` [4] to fit to eigenphase sums and S-matrix elements to extract resonance positions, widths and partial widths as a function of energy or external field.

2.6.1 Low-energy collision properties

MOLSCAT has many features designed to facilitate low-energy scattering calculations.

Scattering lengths and volumes

MOLSCAT can calculate scattering lengths/volumes, which may be complex in the presence of inelastic channels (section 9.10.1). The diagonal S-matrix element in an incoming channel 0 may be written in terms of a complex phase shift η,

$$S_{00} = \exp(2i\eta). \tag{2.13}$$

For a channel with low kinetic energy, the phase shift may be expanded in powers of the incoming wavevector k. The leading term is proportional to k for s-wave scattering and k^3 for p-wave scattering. For higher values of L the leading power depends on the form of the long-range potential: e.g., for a potential that varies as $-C_6R^{-6}$ at long range, the leading term is proportional to k^4 for all $L > 1$. MOLSCAT calculates scattering lengths for $L = 0$ ($n = 1$), volumes for $L = 1$ ($n = 3$) and hypervolumes for $L > 1$ ($n = 4$) using the formula [14]

$$a_L(k) = \frac{-\tan\eta}{k^n} = \frac{1}{ik^n}\left(\frac{1 - S_{00}}{1 + S_{00}}\right). \tag{2.14}$$

It should be emphasised that Eq. 2.14 is an identity, so that this is a far more general approach than that used by some other programs that take the limit of other (often more complicated) functions as $k \to 0$. The scattering length and volume defined by Eq. 2.14 become independent of k at sufficiently low k.

Characterisation of zero-energy Feshbach resonances

MOLSCAT can converge on and characterise the zero-energy Feshbach resonances that appear in the scattering length as a function of external fields [13] (section 9.10.3). It can do this both for resonances in elastic scattering, where the scattering length has a simple pole characterised by its position and width and the background scattering length, and in inelastic scattering, where the resonant behaviour is more complex and requires additional parameters [14].

2.6.2 Infinite-order sudden approximation

MOLSCAT incorporates code for calculating degeneracy-averaged and line-shape cross sections within the infinite-order sudden (IOS) ansatz. In this formulation, S matrices are calculated

from propagations carried out at fixed molecular orientations. The cross sections are written in terms of sums of products of dynamical factors Q and spectroscopic coefficients F. The dynamical factors contain all the information about the collision dynamics; they are defined as integrals over the fixed-orientation S matrices, which are evaluated by numerical quadrature. The spectroscopic coefficients contain information about rotor levels and angular momentum coupling. The IOS code has not been used much in recent years, but is retained for backwards compatibility. It does not provide low-energy features such as scattering lengths.

2.7 Results of bound-state calculations

The quantum-mechanical bound-state problem can be formulated as a set of coupled differential equations similar to those encountered in scattering theory. The difference between the two cases is in the boundary conditions that must be applied. True bound states exist only at energies where all asymptotic channels are energetically closed, $E < E_{\mathrm{intl},i}$ for all i. Under these circumstances the bound-state wavefunction $\boldsymbol{\psi}(R)$ is a column vector of order N that must approach zero in the classically forbidden regions at both short range, $R \to 0$, and long range, $R \to \infty$.

Continuously differentiable solutions of the coupled equations that satisfy the boundary conditions at both ends exist only at specific energies E_n. These are the eigenvalues of the total Hamiltonian (2.1); we refer to them (somewhat loosely) as the eigenvalues of the coupled equations, to distinguish them from eigenvalues of other operators that also enter the discussion below.

Wavefunction matrices $\boldsymbol{\Psi}(R)$ that satisfy the boundary conditions in *one* of the classically forbidden regions exist at any energy. We designate these $\boldsymbol{\Psi}^{+}(R)$ for the solution propagated outwards from short range and $\boldsymbol{\Psi}^{-}(R)$ for the solution propagated inwards from long range. The corresponding log-derivative matrices are $\mathbf{Y}^{+}(R)$ and $\mathbf{Y}^{-}(R)$.

It is convenient to choose a matching distance R_{match} where the outwards and inwards solutions are compared. A solution vector that is continuous at R_{match} must satisfy

$$\boldsymbol{\psi}(R_{\mathrm{match}}) = \boldsymbol{\psi}^{+}(R_{\mathrm{match}}) = \boldsymbol{\psi}^{-}(R_{\mathrm{match}}). \tag{2.15}$$

Since the derivatives of the outwards and inwards solutions must match too, we require that

$$\frac{d}{dR}\boldsymbol{\psi}^{+}(R_{\mathrm{match}}) = \frac{d}{dR}\boldsymbol{\psi}^{-}(R_{\mathrm{match}}) \tag{2.16}$$

so that

$$\mathbf{Y}^{+}(R_{\mathrm{match}})\boldsymbol{\psi}(R_{\mathrm{match}}) = \mathbf{Y}^{-}(R_{\mathrm{match}})\boldsymbol{\psi}(R_{\mathrm{match}}). \tag{2.17}$$

Equivalently,

$$\left[\mathbf{Y}^{+}(R_{\mathrm{match}}) - \mathbf{Y}^{-}(R_{\mathrm{match}})\right]\boldsymbol{\psi}(R_{\mathrm{match}}) = 0, \tag{2.18}$$

so that the wavefunction vector $\boldsymbol{\psi}(R_{\mathrm{match}})$ is an eigenvector of the log-derivative matching matrix, $\Delta\mathbf{Y} = [\mathbf{Y}^{+}(R_{\mathrm{match}}) - \mathbf{Y}^{-}(R_{\mathrm{match}})]$, with eigenvalue zero [7].

For each J_{tot} and symmetry block, BOUND propagates log-derivative matrices to a matching point R_{match}, both outwards from the classically forbidden region at short range (or from $R = 0$) and inwards from the classically forbidden region at long range. At each energy E, it calculates the multichannel node count, defined as the number of zeros of $\boldsymbol{\psi}(R)$ between R_{min} and R_{max}. Johnson [15] showed that this is equal to the number of states that lie below E. It may be calculated as a simple byproduct of the propagations and the matching matrix. BOUND uses the node count to determine the number of states in the specified range, and then uses bisection to identify energy windows that contain exactly one state. In each such window, it uses a combination of bisection and the Van Wijngaarden-Dekker-Brent algorithm [16] to converge on the energy where an eigenvalue of the log-derivative matching matrix $\boldsymbol{\Delta}\mathbf{Y}$ is zero. This is the energy of a state. The program extracts the local wavefunction vector $\boldsymbol{\psi}(R_{\mathrm{match}})$, and optionally calculates the complete bound-state wavefunction $\boldsymbol{\psi}(R)$ using the method of Thornley and Hutson [17].

FIELD operates in a very similar manner to locate states as a function of external field at fixed energy (or energy fixed with respect to a field-dependent threshold energy). The one significant difference is that the multichannel node count is not guaranteed to be a monotonic function of field, and it is in principle possible to miss pairs of states that cross the chosen energy in opposite directions as a function of field. In practice this seldom happens.

Chapter 3

Using the programs: a basic guide

3.1 Prerequisites

The user must always specify:

- the type of system, e.g., atom + linear rigid rotor, diatom + diatom, etc.;
- any dynamical approximations to be applied in setting up the coupled equations;
- the energy levels of the interacting particles, or atomic/molecular constants that describe them;
- the basis set $\{\Phi_j(\xi_{\rm intl})\}$ to be used for the internal coordinates $\xi_{\rm intl}$ (everything except the interparticle separation R).
- the interaction potential $V(R, \xi_{\rm intl})$;
- the reduced mass.

The programs use this information to set up the required coupled equations.

In addition, the user must choose:

- the propagator(s) to be used to solve the coupled equations;
- parameters to control the energies (and, if appropriate, external fields) at which the coupled equations are to be solved;
- parameters to control the range of interparticle distance over which the coupled equations are to be solved, and the propagation step size;
- parameters to control optional processing, such as calculating line-shape cross sections or converging on scattering resonances;
- parameters to control the level of printed output and optional additional output and scratch files.

Parameters are input to the program from a plain-text file in namelist format as described below. The types of system supported are summarised in section 3.3 and the numerical methods available for solving the coupled differential equations are summarised in section 3.5.

3.2 Input data format

The main input file for any of the programs is read on unit 5 (standard input), and consists of several blocks of namelist data in the following order.

`&INPUT` is read in subroutine `DRIVER`, and provides overall control of the calculation: reduced mass, collision or binding energies, external fields, total angular momenta, choice of propagators, propagation ranges and step sizes, specification of optional calculations (line-shape cross sections, resonance characterisation, etc.), output files and print control. The available input parameters are described in chapters 6, 7, 8, 9, 10, 12 and 13. Throughout the remainder of this document (with the exception of the glossary in chapter 11) items in `&INPUT` are coloured red.

`&BASIS` is read in entry `BASIN` of subroutine `BASE`, and describes the type of system, atomic and molecular parameters, basis set, and dynamical approximations. The available input parameters are described in chapter 4. Throughout the remainder of this document (with the exception of the glossary in chapter 11) items in `&BASIS` are coloured blue.

`&POTL` is read in the initialisation call of the general-purpose version of subroutine `POTENL`, and specifies the interaction potential to be used. The available input parameters are described in chapter 5. Throughout the remainder of this document (with the exception of the glossary in chapter 11) items in `&POTL` are coloured green.

For some cases it may be desirable to substitute a special-purpose `POTENL` routine, which does not necessarily read namelist `&POTL`. The specification of `POTENL` is given in chapter 16 for use by those who wish to substitute their own routine for the general-purpose version.

All the namelist items are listed in the glossary in chapter 11, together with their default values, a brief description and references to where the reader can find more information.

In addition, there may be other blocks of input data required by user-supplied subroutines, which must be inserted in the correct place between or after the namelist blocks listed above. In particular, many plug-in basis-set suites accessed through `ITYPE` = 9 utilise an additional namelist block named `&BASIS9`. These include the suite for interaction of two alkali-metal atoms, described in chapter 17.

Each namelist block in the input file must start with `&<name>` or `$<name>`, where `<name>` is `INPUT`, `BASIS` or `POTL` as appropriate, and should be terminated by a slash. Between these delimiters, the namelist input data consist of entries of the form `KEYWORD =value` or `KEYWORD =list of values`, where `KEYWORD` is the variable or array name. Values can be separated by commas, spaces, tabs, or end-of-line. A value can be repeated by preceding it with a multiplier and the `*` character. The Fortran 90 standard specifies that all entries (including the `$<name>` and the slash) should begin in column 2 or later, but most common compilers are more flexible. Most compilers allow the use of comments; these must be preceded by an exclamation mark and continue to the end of the line.

Most of the parameters have sensible default values (which are given in this document); these are used if the parameter is not included in the namelist block.

3.3 Interaction types

The term *interaction type* describes the types of the interacting monomers and any dynamical approximations to be applied. It was originally *collision type* in MOLSCAT, but has been generalised here to include pairs of monomers that form bound states.

The programs can perform close-coupling calculations (with no dynamical approximations) for the following interaction types:

1. Atom + linear rigid rotor [18];
2. Atom + vibrating diatom (rotationally and/or vibrationally inelastic) with interaction potentials independent of diatom rotational state [19];
3. Linear rigid rotor + linear rigid rotor [20, 21, 22];
4. Asymmetric top + linear molecule [23]
5. Atom + symmetric top (also handles near-symmetric tops and linear molecules with vibrational angular momentum) [24, 25];
6. Atom + asymmetric top [24] (also handles spherical tops [26]);
7. Atom + vibrating diatom (rotationally and/or vibrationally inelastic) with interaction potentials dependent on diatom rotational state [27];
8. Atom + rigid corrugated surface: diffractive (elastic) scattering [28, 29]. At present, the code is restricted to centrosymmetric lattices, for which the potential matrices are real;
9. Interaction type specified in a plug-in basis-set suite (which may be user-supplied). A substantial number of these plug-in suites exist. This release includes two representative examples:
   - Structureless atom + $^3\Sigma$ molecule in a magnetic field, demonstrated for Mg + NH;
   - Two alkali-metal atoms, including hyperfine coupling and magnetic field, demonstrated for $^{85}Rb_2$.

The quantities that control the quantum states included in the basis set and the corresponding internal energies of the interacting partners are specified in namelist `&BASIS`.

The computer time required to solve a set of N coupled equations is approximately proportional to N^3. The practical limit on N is from a few hundred to several thousand, depending on the speed of the computer and the amount of memory available. The basis sets necessary for converged close-coupling calculations may easily exceed this limit as scattering energies increase or rotational constants decrease, particularly for interaction types other than the very simplest. However, the programs also provide various approximate (decoupling) methods that reduce the number of coupled equations. The methods supported at present are:

- Effective potential approximation [30] (seldom used nowadays);
- Coupled-states (centrifugal sudden) approximation [31];
- Decoupled L-dominant approximation [32, 33] (seldom used nowadays);
- Infinite-order sudden approximation (MOLSCAT only) [34, 35, 25].

Not all these approximations are supported for all interaction types, though the most common ones are. The programs print a warning message and exit if an unsupported approximation is requested.

3.4 Interaction potential

The programs call a routine (named `POTENL`) to evaluate the radial potential coefficients $v_\Lambda(R)$ of Eq. 2.7 that describe the interaction potential from information provided by the user. The general-purpose version of `POTENL` obtains information about the interaction potential either from namelist `&POTL`, or from user-supplied routines that may either provide the radial potential coefficients directly, or (for some interaction types) may provide values of the potential at specified points for expansion within `POTENL`. More information about this given in chapter 5.

3.5 Propagators

The coupled equations may be solved using any one of several methods:

de Vogelaere propagator (DV) [36]: This propagates the wavefunction explicitly, but is much slower than more modern methods, especially for large reduced masses or high scattering energies. It is not recommended except for special purposes.

R-matrix propagator (RMAT) [37]: This is a stable method that works in a quasiadiabatic basis. It has relatively poor step-size convergence properties, and has largely been superseded by the log-derivative propagators. It is not recommended except for special purposes.

Log-derivative propagator of Johnson (LDJ) [38, 39]: This is a very stable propagator. It has largely been superseded by the LDMD propagator, but can be useful in occasional cases where that propagator has trouble evaluating node counts.

Diabatic log-derivative method of Manolopoulos (LDMD) [40]: This is a very efficient and stable propagator, especially at short and medium range. It is coded to detect single-channel cases (including IOS cases) automatically and in that case use a more efficient implementation.

Quasiadiabatic log-derivative propagator of Manolopoulos (LDMA) [41, 7]: This is similar to the LDMD propagator, but operates in a quasiadiabatic basis. It offers better accuracy than LDMD for very strongly coupled problems, but is relatively expensive. It is recommended for production runs only for very strongly coupled problems. However, it is also useful when setting up a new system, because it can output eigenvalues of the interaction matrix at specific distances (adiabats) and nonadiabatic couplings between the adiabatic states.

Symplectic log-derivative propagators of Manolopoulos and Gray (LDMG) [9]: This offers a choice of 4th-order or 5th-order symplectic propagators. These are 1.5 to 3 times more expensive per step than the LDMD and LDJ propagators, but can have smaller errors for a given step size. They are often the most efficient choice when high precision is required.

AIRY propagator: This is the AIRY log-derivative propagator of Alexander [42] as reformulated by Alexander and Manolopoulos [43]. It uses a quasiadiabatic basis with a linear

reference potential (which results in Airy functions as reference solutions). This allows the step size to increase rapidly with separation, so that this propagator is particularly efficient at long range.

VIVS propagator [44]: This is the variable-interval variable-step method of Parker *et al.* and is intended for use at long range. It is sometimes very efficient, but the interval size is limited when there are deeply closed channels, so that it is not efficient at long range in such cases. Control of it is considerably more complicated than for other propagators, and it has largely been superseded by the AIRY propagator.

WKB semiclassical integration using Gauss-Mehler quadrature [45]: This is not a true propagator and can be used only for single-channel problems.

In BOUND and FIELD, only log-derivative propagators are implemented.

All these propagators have options that allow them to use interaction matrices stored at the first total energy when doing calculations at subsequent energies. For the RMAT, VIVS, and log-derivative propagators, some of the remaining work is also avoided at subsequent energies, so that in special circumstances they may cost only 30% as much CPU time as the first energy.

Gordon's propagator, which was available in early versions of MOLSCAT, is not implemented in version 2020.0.

Recommendation:

For applications that do not need high precision, the LDMD propagator provides a good balance between stability and efficiency at short and medium range. When high precision is required, the LDMG propagator may be a better choice.

For problems that do not require long-range propagation, the LDMD or LDMG propagator may be used on its own. When propagation to very long range is required, it is usually best to combine it with the AIRY propagator for the long-range part of the propagation.

The remaining propagators should be used only for special purposes by expert users.

3.6 Overview of main input file

The main input file specifies the calculation required, and the associated tasks may be grouped roughly as follows:

3.6.1 Scattering and bound-state calculations

1. Specify the interaction type; see the preamble in chapter 4
2. Specify appropriate loops over total angular momentum and/or other symmetries; see section 4.9
3. Construct a basis set appropriate to the interacting partners; see the rest of chapter 4
4. Construct a potential expansion appropriate to the interacting partners; see chapter 5
5. Specify energies and external fields; see chapters 6 and 7

6. Choose propagator(s) and specify propagation ranges and step sizes; see chapter 8

3.6.2 Scattering calculations

If the corresponding option is requested:

7. Control automated testing of convergence of S-matrix elements with respect to propagation parameters; see section 9.5
8. Specify spectroscopic lines for line-shape cross sections; see section 9.7
9. Control searches for energy-dependent resonances; see section 9.9
10. Control convergence on field-dependent resonances; see section 9.10.3
11. Specify the incoming channel for effective-range calculations; see section 9.10.5
12. Specify the incoming channel for a wavefunction calculation; see section 9.11

3.6.3 Bound-state calculations

If the corresponding option is requested:

7. Specify expectation values to be calculated (without wavefunctions) (BOUND only); see section 10.5
8. Use automated testing of convergence of bound-state energies and expectation values with respect to propagation parameters (BOUND only); see section 10.6
9. Specify a wavefunction calculation; see section 10.4.

3.7 Units of mass, length and energy

By default, the programs operate with masses in unified atomic mass units (Daltons), lengths in Å (1 Å = 10^{-10} m) and energies E expressed as wavenumbers E/hc in cm^{-1}. However, all these may be altered using the variables MUNIT, RUNIT and EUNIT, which give values for the required units in Daltons, Å and cm^{-1} respectively.

Commonly used energy units may be selected with the integer variable EUNITS (default 1) in place of EUNIT. The allowed values are

EUNITS = **1** cm^{-1}
EUNITS = **2** Kelvin
EUNITS = **3** MHz
EUNITS = **4** GHz
EUNITS = **5** eV
EUNITS = **6** erg
EUNITS = **7** hartree (atomic unit of energy)
EUNITS = **8** kJ/mol
EUNITS = **9** kcal/mol

The value in EUNIT is used only if EUNITS is zero.

3.8 Examples for MOLSCAT

3.8.1 Interpretation of the complete output for a model system

The input file `molscat-basic1.input` set up a small calculation of cross sections for collisions between an atom and a rigid rotor as follows:

```
 &INPUT
    LABEL  = 'model system: ITYPE=1',
    URED   =  20.0,
    IPRINT =   1,   ISIGPR =    2,
```

The print level IPRINT = 1 specifies minimal output, and ISIGPR = 1 specifies output of state-to-state cross sections.

```
 ---- MOLSCAT ------ MOLSCAT ------ MOLSCAT ------ MOLSCAT ------ MOLSCAT ---
|                                                                             |
|               Non-reactive quantum scattering calculations                  |
|                    on atomic and molecular collisions                       |
|                                                                             |
|             Copyright (C) 2020 J. M. Hutson & C. R. Le Sueur                |
|                                                                             |
|                                 Version 2020.0                              |
|                                                                             |
|                       Run on xx Xxx xxxx   at xx:xx:xx                      |
|                                                                             |
```

```
---- MOLSCAT ------ MOLSCAT ------ MOLSCAT ------ MOLSCAT ------ MOLSCAT ---


This program is free software: you can redistribute it and/or modify it under
the terms of the GNU General Public License, version 3, as published by
the Free Software Foundation.

Publications resulting from the use of this program should cite both
the version of the program used:
J. M. Hutson & C. R. Le Sueur, MOLSCAT computer code 2020.0
and the published paper:
J. M. Hutson & C. R. Le Sueur, Comput. Phys. Commun. 241, pp 9-18 (2019).


USING CODATA 2018 RECOMMENDED VALUES OF FUNDAMENTAL PHYSICAL CONSTANTS

MEMORY ALLOCATED TO MAIN WORKING ARRAY IS   1000000 (8-BYTE) WORDS (    7.63 MB)
2 INTEGERS CAN BE STORED IN EACH WORD.

PRINT LEVEL (IPRINT) =  1     OTHER PRINT CONTROLS  ISIGPR = 2

REDUCED MASS FOR INTERACTION =  20.000000000 ATOMIC MASS UNITS (DALTONS)
```

The parameters input in namelist `&BASIS` specify the interaction type and energy levels of the colliding partners:

```
ITYPE  = 1,  BE     = 30.0,
NLEVEL = 4,  JLEVEL =  0, 2, 4, 6,
```

resulting in the following initialisation output:

```
INTERACTION TYPE IS    LINEAR RIGID ROTOR  -  ATOM.

MOLECULAR QUANTUM NUMBERS TAKEN FROM JLEVEL INPUT.  NLEVEL =  4

ENERGY LEVELS OBTAINED FROM B(E) =   30.000000

QUANTUM NUMBERS FOR INTERACTING PAIR:
EACH PAIR STATE IS LABELLED BY  1 QUANTUM NUMBER
EACH CHANNEL FUNCTION IS FORMED BY COMBINING A PAIR STATE WITH A VALUE OF L.
THE RESULTING BASIS SET IS ASYMPTOTICALLY DIAGONAL.

PAIR STATE     PAIR STATE QUANTUM NUMBERS    PAIR LEVEL     PAIR ENERGY (CM-1)
              ---------        J ---------
      1                        0               1            0.0000000
      2                        2               2          180.0000000
      3                        4               3          600.0000000
      4                        6               4         1260.0000000
```

The parameters input in namelist `&POTL` items control the interaction potential. For `ITYPE` $= 1$, the potential is expanded in Legendre polynomials $P_\lambda(\cos\theta)$. In this calculation each of the `MXLAM` $= 2$ expansion terms (corresponding to $\lambda = 0$ and 2) consists of one or two inverse-power expressions:

```
     MXLAM = 2,   LAMBDA =   0,          2,
                  NTERM  =   2,          1,
                  NPOWER = -12,   -6,   -6,
                  A      =   1.0, -2.0, -0.2,
```

```
GENERAL-PURPOSE POTENL ROUTINE (MAY 18)

ANGULAR DEPENDENCE OF POTENTIAL EXPANDED IN TERMS OF
LEGENDRE POLYNOMIALS, P(LAMBDA).

INTERACTION POTENTIAL FOR EXPANSION TERM NUMBER   1
WHICH HAS LAMBDA =   0

               1.00000000E+00 * R **-12
              -2.00000000E+00 * R ** -6

INTERACTION POTENTIAL FOR EXPANSION TERM NUMBER   2
WHICH HAS LAMBDA =   2

              -2.00000000E-01 * R ** -6

POTENL PROCESSING FINISHED.
```

The units used for length throughout the run and for interaction energies returned from the potential routine are specified as RM and EPSIL in namelist &POTL:

```
     RM    = 3.5, EPSIL  =  50.0,
```

resulting in the following output:

```
POTENTIAL RETURNED IN UNITS OF EPSIL  =  50.000000     CM-1
      CODED WITH R IN UNITS OF RM     =  3.5000000     ANGSTROM

ALL LENGTHS ARE IN UNITS OF RM (  3.50000000 ANGSTROM ) UNLESS OTHERWISE STATED

INTERACTION MATRIX USES   2 BLOCKS OF VL ARRAY FOR R-DEPENDENT TERMS IN POTENTIAL
```

The following items in &INPUT control the propagation:

```
     RMIN   =   0.5, RMAX   =   20.0,
     IPROPS =   6,   DR     =    0.001,
```

and result in the following output at the initialisation stage:

```
PROPAGATION METHODS FOR COUPLED EQUATIONS SPECIFIED BY IPROPS =  6

COUPLED EQUATIONS WILL BE PROPAGATED OUTWARDS IN 1 SEGMENT

PROPAGATION RANGE IS CONTROLLED BY VARIABLES RMIN AND RMAX, WITH INPUT VALUES
RMIN =  0.5000     RMAX =   20.00
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
SEGMENT 1 WILL BE PROPAGATED OUTWARDS

FROM RMIN CHOSEN USING IRMSET =  9 TO RMAX =     20.00

COUPLED EQUATIONS SOLVED BY DIABATIC MODIFIED LOG-DERIVATIVE PROPAGATOR OF MANOLOPOULOS

PROPAGATION STEP SIZE DETERMINED USING DR =  1.000E-03
STEP SIZE CONSTANT THROUGHOUT RANGE
STEP SIZE MAY BE ADJUSTED SLIGHTLY SO THAT RANGE IS A WHOLE NUMBER OF STEPS

LOG-DERIVATIVE MATRIX INITIALISED IN THE LOCAL EIGENBASIS AT RMIN
LOCALLY CLOSED CHANNELS INITIALISED WITH A WKB BOUNDARY CONDITION
```

Energies are assumed to be in cm^{-1} by default, and the input file specifies that only 1 energy is to be used

```
NNRG   =   1,   ENERGY = 1250.0,
```

```
INPUT ENERGIES ASSUMED TO BE IN UNITS OF CM-1 BY DEFAULT.

CALCULATIONS WILL BE PERFORMED FOR    1 ENERGY
ENERGY    1           =   1250.000000     CM-1
```

The following items in `&INPUT` control the values of total angular momentum for which calculations are executed:

```
JTOTL  =  10,   JTOTU  =   20,     JSTEP = 10,
```

resulting in

```
TOTAL ANGULAR MOMENTUM JTOT RUNS FROM  10  TO     20  IN STEPS OF  10

EACH JTOT IS SPLIT INTO A MAXIMUM OF   2 SYMMETRY BLOCKS
```

and this concludes the initialisation procedures.

The program then proceeds to do 4 propagations (2 values of J_{tot}, each of which is factorised into 2 symmetry blocks), which are summarised:

```
============================================== model system: ITYPE=1 ==============================================

FOR JTOT =   10, SYMMETRY BLOCK =  1, ENERGY(  1) =  1250.000    : MAX DIAG & OFF-DIAG =  9.88E-02 &   1.16E-06

FOR JTOT =   10, SYMMETRY BLOCK =  2, ENERGY(  1) =  1250.000    : MAX DIAG & OFF-DIAG =  1.75E-01 &   1.29E-03

FOR JTOT =   20, SYMMETRY BLOCK =  1, ENERGY(  1) =  1250.000    : MAX DIAG & OFF-DIAG =  1.88E-01 &   1.95E-06

FOR JTOT =   20, SYMMETRY BLOCK =  2, ENERGY(  1) =  1250.000    : MAX DIAG & OFF-DIAG =  1.68E-01 &   2.43E-03
```

The program calculates an S matrix from each propagation, which is printed if IPRINT ≥ 11 (but not here). It uses the S matrices to calculate partial cross sections, which are printed if IPRINT ≥ 5 (but not here). At the end of the calculation, the program prints the degeneracy-averaged state-to-state integral cross sections. These are from initial levels `I` to final levels `F`; the level energies are given here, and the corresponding quantum numbers may be obtained from the list above.

```
LEVEL   4 WITH ENERGY   1260.000000000000 IS NEVER OPEN

STATE-TO-STATE INTEGRAL CROSS SECTIONS IN ANGSTROM**2 BETWEEN    3 LEVELS WITH THRESHOLD ENERGIES (IN CM-1):

  1     0.000000000000
  2   180.000000000000
  3   600.000000000000

*** N.B. CROSS SECTIONS HAVE BEEN MULTIPLIED BY 10.0 TO ACCOUNT FOR JSTEP

 ENERGY (CM-1)    JTOTL  JSTEP  JTOTU      F    I         SIG(F,I)
 1250.000000         10     10     20      1    1          1.81057
 1250.000000         10     10     20      2    1         3.722327E-02
 1250.000000         10     10     20      3    1         6.542671E-07

 1250.000000         10     10     20      1    2         8.697026E-03
 1250.000000         10     10     20      2    2          3.00836
 1250.000000         10     10     20      3    2         8.161625E-05

 1250.000000         10     10     20      1    3         1.398007E-07
 1250.000000         10     10     20      2    3         7.464050E-05
 1250.000000         10     10     20      3    3          5.24476
```

and the total inelastic integral cross section from each initial level:

```
TOTAL INELASTIC INTEGRAL CROSS SECTIONS IN ANGSTROM**2 FROM LEVEL
 3.72239E-02                                                  1
 8.77864E-03                                                  2
 7.47803E-05                                                  3
```

The output terminates with a footer message:

```
 ---- MOLSCAT ------ MOLSCAT ------ MOLSCAT ------ MOLSCAT ------ MOLSCAT ---
|                                                                              |
|               Non-reactive quantum scattering calculations                   |
|                    on atomic and molecular collisions                        |
|                                                                              |
|             Copyright (C) 2020 J. M. Hutson & C. R. Le Sueur                 |
|                                                                              |
|                                 Version 2020.0                               |
|                                                                              |
|                     This run used       xxxx cpu secs and                    |
|                  1279 of the allocated   1000000 words of storage            |
|                                                                              |
 ---- MOLSCAT ------ MOLSCAT ------ MOLSCAT ------ MOLSCAT ------ MOLSCAT ---
```

3.8.2 Higher print level

The following example (whose output is not listed in full) illustrates the output at a higher print level, IPRINT = 11, for a more sophisticated case involving collisions of two rigid rotors, specifically para-H_2 colliding with ortho-H_2. It also uses a realistic potential provided by a VRTP routine.

Only sections of the output file that differ in important ways from section 3.8.1 are described here.

`molscat-basic2.input` contains the input data:

```
&INPUT
   LABEL  = 'p-H2 + o-H2: potential of Zarur and Rabitz supplied by VRTP',
   URED   =  1.00794,
   IPRINT = 11,    ISIGPR =   1,
   RMIN   =  0.43, RMID   =   2.0, RMAX   = 20.0,  IRXSET =  1,
   IPROPS =  6,    IPROPL =   9,   STEPS  = 15.0,
   JTOTL  =  6,    JTOTU  =   6,
   NNRG   =  1,    ENERGY = 700.0,
/

&BASIS
   ITYPE  = 3,
   BE     = 2*59.067,
   NLEVEL = 3,   JLEVEL = 0,1,  0,3,  2,1,
/

&POTL
   MXLAM  = 4,      NTERM  = 4*-1,
                    LAMBDA = 0,0,0,  2,0,2,  0,2,2,  2,2,4,
   LVRTP  = .TRUE.,
```

```
 /
```

The propagation is carried out in 2 parts, with different propagators at short and long range. This produces the following output:

```
PROPAGATION METHODS FOR COUPLED EQUATIONS SPECIFIED BY IPROPS =  6 AND IPROPL =  9

COUPLED EQUATIONS WILL BE PROPAGATED OUTWARDS IN 2 SEGMENTS

PROPAGATION RANGES ARE CONTROLLED BY VARIABLES RMIN, RMID AND RMAX, WITH INPUT VALUES
RMIN =  0.4300     RMID =   2.000     RMAX =   20.00
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
SEGMENT 1 WILL BE PROPAGATED OUTWARDS

FROM RMIN CHOSEN USING IRMSET =  9 TO RMID =      2.00

COUPLED EQUATIONS SOLVED BY DIABATIC MODIFIED LOG-DERIVATIVE PROPAGATOR OF MANOLOPOULOS

PROPAGATION STEP SIZE DETERMINED USING STEP =   15.0     (PER WAVELENGTH)
STEP SIZE CONSTANT THROUGHOUT RANGE
STEP SIZE MAY BE ADJUSTED SLIGHTLY SO THAT RANGE IS A WHOLE NUMBER OF STEPS

LOG-DERIVATIVE MATRIX INITIALISED IN THE LOCAL EIGENBASIS AT RMIN
LOCALLY CLOSED CHANNELS INITIALISED WITH A WKB BOUNDARY CONDITION
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
SEGMENT 2 WILL BE PROPAGATED OUTWARDS

FROM RMID =      2.00 TO WHICHEVER IS LARGER OF
OUTERMOST CENTRIFUGAL TURNING POINT IN OPEN CHANNELS, AND RMAX, WHICH =     20.00

COUPLED EQUATIONS SOLVED BY VARIABLE-STEP AIRY PROPAGATOR.
PUBLICATIONS RESULTING FROM THE USE OF THIS PROPAGATOR SHOULD REFERENCE
M. H. ALEXANDER AND D. E. MANOLOPOULOS,  J. CHEM. PHYS. 86, 2044 (1987).

PROPAGATION STEP SIZE DETERMINED USING STEP =   15.0     (PER WAVELENGTH)
STEP SIZES ADJUSTED TO MAINTAIN APPROXIMATE ACCURACY VIA PERTURBATION THEORY
WITH TOLHI =  1.00E-04 AND POWR =   3.0
```

The basis set is described in terms of pair levels and pair states for the molecule-molecule system, as described in section 4.2. The following output lists the pair state quantum numbers and corresponding pair levels and pair energies. In this case there are several pair states arising from one of the pair levels.

```
INTERACTION TYPE IS    LINEAR ROTOR - LINEAR ROTOR.

PAIR LEVEL QUANTUM NUMBERS TAKEN FROM JLEVEL INPUT.  NLEVEL =  3

ENERGY LEVELS OF ROTOR 1 OBTAINED FROM B(E) =   59.067000

ENERGY LEVELS OF ROTOR 2 OBTAINED FROM B(E) =   59.067000

QUANTUM NUMBERS FOR INTERACTING PAIR:
EACH PAIR STATE IS LABELLED BY  3 QUANTUM NUMBERS
```

```
EACH CHANNEL FUNCTION IS FORMED BY COMBINING A PAIR STATE WITH A VALUE OF L.
THE RESULTING BASIS SET IS ASYMPTOTICALLY DIAGONAL.

PAIR STATE    - PAIR STATE QUANTUM NUMBERS -   PAIR LEVEL     PAIR ENERGY (CM-1)
                    J1        J2       J12
      1              0         1         1       1           118.1340000
      2              0         3         3       2           708.8040000
      3              2         1         1       3           472.5360000
      4              2         1         2       3           472.5360000
      5              2         1         3       3           472.5360000
```

The general-purpose potential routine produces the following output, which describes the symmetries of the colliding molecules (both homonuclear), and the expansion used for the interaction potential. LVRTP = .TRUE. specifies that the potential coefficients are to be obtained by quadrature, with the potential at the quadrature points evaluated by a user-supplied routine VRTP.

```
GENERAL-PURPOSE POTENL ROUTINE (MAY 18)

UNEXPANDED POTENTIAL IS OBTAINED FROM VRTP ROUTINE.

A SUITABLE VRTP ROUTINE MUST BE SUPPLIED.

[OUTPUT FROM INITIALISATION OF SUPPLIED VRTP ROUTINE]

IHOMO  = 2 SPECIFIES HOMONUCLEAR SYMMETRY FOR ROTOR 1.
IHOMO2 = 2 SPECIFIES HOMONUCLEAR SYMMETRY FOR ROTOR 2.
USING   3-POINT QUADRATURE FOR THETA-1
HOMONUCLEAR SYMMETRY: ONLY HALF OF THE THETA-1 POINTS WILL BE USED
USING   3-POINT QUADRATURE FOR THETA-2
HOMONUCLEAR MOLECULE 2: ONLY HALF OF THE THETA-2 POINTS WILL BE USED
USING   3-POINT QUADRATURE FOR PHI

ANGULAR DEPENDENCE OF POTENTIAL EXPANDED IN TERMS OF
CONTRACTED NORMALISED SPHERICAL HARMONICS,
SUM(M1,M2,M) C(L1,M1,L2,M2,L,M) Y(L1,M1) Y(L2,M2) Y(L,M)
SEE GREEN, J. CHEM. PHYS. 62, 2271 (1975)

INTERACTION POTENTIAL FOR EXPANSION TERM NUMBER   1
WHICH HAS LAM1 =   0,  LAM2 =   0,  LAM =   0

INTERACTION POTENTIAL FOR EXPANSION TERM NUMBER   2
WHICH HAS LAM1 =   2,  LAM2 =   0,  LAM =   2

INTERACTION POTENTIAL FOR EXPANSION TERM NUMBER   3
WHICH HAS LAM1 =   0,  LAM2 =   2,  LAM =   2

INTERACTION POTENTIAL FOR EXPANSION TERM NUMBER   4
WHICH HAS LAM1 =   2,  LAM2 =   2,  LAM =   4

POTENL PROCESSING FINISHED.

POTENTIAL RETURNED IN UNITS OF EPSIL  =  24.170000     CM-1
      CODED WITH R IN UNITS OF RM     =  3.4900000     ANGSTROM
```

```
ALL LENGTHS ARE IN UNITS OF RM (  3.49000000 ANGSTROM ) UNLESS OTHERWISE STATED

INTERACTION MATRIX USES   4 BLOCKS OF VL ARRAY FOR R-DEPENDENT TERMS IN POTENTIAL
```

The program now enters a loop over total angular momentum JTOT and symmetry block IBLOCK. The first time through the loop is for JTOT = 6 and IBLOCK = 1. The output for this begins with a list of the channels, describing their relationship to the pair states above:

```
***************************  ANGULAR MOMENTUM JTOT  =   6   AND SYMMETRY BLOCK  =    1  ***************************

CPL3 (JAN 93).  JTOT-INDEPENDENT PARTS OF COUPLING MATRIX STORED.  NSTATE, MXLAM, IEX =   5   4   0
               REQUIRED AND AVAILABLE STORAGE =       60   999188

CHANNEL FUNCTION LIST:

EACH CHANNEL FUNCTION IS FORMED BY COMBINING A PAIR STATE WITH A VALUE OF L.

CHANNEL  PAIR STATE   - PAIR STATE QUANTUM NUMBERS -          L        PAIR LEVEL   PAIR ENERGY (CM-1)
                           J1        J2       J12
      1          2          0         3         3            4             2       708.8040000
      2          4          2         1         2            4             3       472.5360000
      3          5          2         1         3            4             3       472.5360000
      4          1          0         1         1            6             1       118.1340000
      5          2          0         3         3            6             2       708.8040000
      6          3          2         1         1            6             3       472.5360000
      7          4          2         1         2            6             3       472.5360000
      8          5          2         1         3            6             3       472.5360000
      9          2          0         3         3            8             2       708.8040000
     10          4          2         1         2            8             3       472.5360000
     11          5          2         1         3            8             3       472.5360000
```

IRMSET > 0 (default 9) specifies that the program should locate a suitable value of $R_{\min}$ as described in section 8.5. This produces output as follows.

```
INNER CLASSICAL TURNING POINT AT R =  0.7344
RADIAL PROPAGATION WILL START AT R =  0.4242
```

The program now enters a loop over energy. For each energy, it prints the range and number of steps taken for each propagator used:

```
MDPROP. LOG DERIVATIVE MATRIX PROPAGATED FROM       0.4242  TO  2.0000      IN    155  STEPS.

AIRPRP. LOG DERIVATIVE MATRIX PROPAGATED FROM       2.0000  TO  20.000      IN     82  STEPS.
```

After each propagation, the program outputs the list of open channels and the resulting S matrix. The open channels are listed in order of increasing threshold energy (decreasing kinetic energy); the channel index for each open channel may be used to find its quantum numbers in the full channel list above. The S-matrix output is lengthy (64 entries in this case), so is abbreviated here.

```
OPEN CHANNEL   WVEC (1/ANG.)    CHANNEL          L    PAIR LEVEL           PAIR ENERGY (CM-1)
         1    5.89835058E+00         4           6         1            118.134000000000
         2    3.68786851E+00         3           4         3            472.536000000000
         3    3.68786851E+00         2           4         3            472.536000000000
         4    3.68786851E+00         6           6         3            472.536000000000
         5    3.68786851E+00         7           6         3            472.536000000000
         6    3.68786851E+00         8           6         3            472.536000000000
         7    3.68786851E+00        10           8         3            472.536000000000
         8    3.68786851E+00        11           8         3            472.536000000000

ROW  COL       S**2                  PHASE/2PI             RE (S)                 IM (S)
  1    1 9.9281505067014E-001  -1.5519889189038E-001   5.5903016854931E-001  -8.2480320157106E-001
  2    1 7.5823644815925E-004   1.4529426949386E-002   2.7421429470486E-002   2.5103095335835E-003

  [ENTRIES FOR REMAINDER OF 8x8 S MATRIX]

  7    8 1.2342641859248E-003   1.6557774202928E-003   3.5130195009608E-002   3.6549215000926E-004
  8    8 9.9351391981921E-001  -2.4320434980401E-001   4.2546700901928E-002  -9.9584320957748E-001
```

The program outputs the state-to-state partial cross sections. These are the contributions to the corresponding integral cross sections from the current S matrix.

```
* * * * * * * * * *  STATE-TO-STATE PARTIAL CROSS SECTIONS (ANGSTROM**2) FROM LEVEL I TO LEVEL F  * * * * * * * * * *
                     FOR JTOT =   6 AND SYMMETRY BLOCK =   1 AT ENERGY(  1) =  700.0000     CM-1

  F  I =      1             3
  1       3.42292E-01    1.43838E-03
  3       2.81148E-03    3.42742E+00

FOR JTOT =    6, SYMMETRY BLOCK =  1, ENERGY(  1) =  700.0000    : MAX DIAG & OFF-DIAG =  3.43E+00 &   2.81E-03
```

For subsequent values of `JTOT` and/or `IBLOCK`, the program also outputs the values of the integral cross sections accumulated so far; for `JTOT` $= 6$ and `IBLOCK` $= 2$, the corresponding output is

```
* * * * * * * * * *  STATE-TO-STATE PARTIAL CROSS SECTIONS (ANGSTROM**2) FROM LEVEL I TO LEVEL F  * * * * * * * * * *
                     FOR JTOT =   6 AND SYMMETRY BLOCK =   2 AT ENERGY(  1) =  700.0000     CM-1

  F  I =      1             3
  1       1.94114E+00    2.07216E-03
  3       4.05025E-03    2.96884E+00

FOR JTOT =    6, SYMMETRY BLOCK =  2, ENERGY(  1) =  700.0000    : MAX DIAG & OFF-DIAG =  2.97E+00 &   4.05E-03

* * * * * * * * * * STATE-TO-STATE INTEGRAL CROSS SECTIONS: ACCUMULATED FROM JTOT =   6 TO   6  * * * * * * * * * *

  F  I =      1             3
  1       2.28343E+00    3.51054E-03
  3       6.86173E-03    6.39626E+00
```

After all propagations are complete, the program summarises the state-to-state integral cross sections and prints a final footer message as before.

3.9 Example for BOUND

`bound-basic1.input` contains input data for a small calculation of the bound states for the model system in section 3.8.1. The only differences from the corresponding input file for MOLSCAT are that the collision energies in ENERGY are replaced with a range specified by EMIN and EMAX and a matching point RMATCH is specified. The coupled equations are now propagated in two parts: outwards from R_{min} to R_{match} and inwards from R_{max} to R_{match}.

```
PROPAGATION METHODS FOR COUPLED EQUATIONS SPECIFIED BY IPROPS =  6 AND IPROPL =  6

COUPLED EQUATIONS WILL BE PROPAGATED TOWARDS RMATCH IN 2 SEGMENTS

PROPAGATION RANGES ARE CONTROLLED BY VARIABLES RMIN, RMATCH AND RMAX, WITH INPUT VALUES
RMIN =  0.5000     RMATCH =   1.000     RMAX =   20.00
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
SEGMENT 1 WILL BE PROPAGATED OUTWARDS

FROM RMIN =      0.50 TO RMATCH =      1.00

COUPLED EQUATIONS SOLVED BY DIABATIC MODIFIED LOG-DERIVATIVE PROPAGATOR OF MANOLOPOULOS

PROPAGATION STEP SIZE DETERMINED USING DR =  1.000E-03
STEP SIZE CONSTANT THROUGHOUT RANGE
STEP SIZE MAY BE ADJUSTED SLIGHTLY SO THAT RANGE IS A WHOLE NUMBER OF STEPS

LOG-DERIVATIVE MATRIX INITIALISED IN THE LOCAL EIGENBASIS AT RMIN IN THE OUTWARD PROPAGATION PART
LOCALLY CLOSED CHANNELS INITIALISED WITH A WKB BOUNDARY CONDITION
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
SEGMENT 2 WILL BE PROPAGATED  INWARDS

TO RMATCH =      1.00 FROM RMAX =     20.00

COUPLED EQUATIONS SOLVED BY DIABATIC MODIFIED LOG-DERIVATIVE PROPAGATOR OF MANOLOPOULOS

INITIAL STEP SIZE TAKEN FROM SIZE OF FINAL STEP OF SHORT-RANGE PROPAGATION
STEP SIZE CONSTANT THROUGHOUT RANGE
STEP SIZE MAY BE ADJUSTED SLIGHTLY SO THAT RANGE IS A WHOLE NUMBER OF STEPS

LOG-DERIVATIVE MATRIX INITIALISED IN THE LOCAL EIGENBASIS AT RMAX IN THE INWARD PROPAGATION PART
LOCALLY CLOSED CHANNELS INITIALISED WITH A WKB BOUNDARY CONDITION
LOCALLY  OPEN  CHANNELS INITIALISED WITH THE VALUE  0.000E+00
```

The matrix of the difference between the two resulting log-derivative matrices at R_{match} is singular if a bound state exists at that energy. A singular matrix has at least one eigenvalue that is zero, so BOUND performs a 1D searches for and then converges on zero-valued eigenvalues of the matching matrix. The input file requests location of bound states with binding energies between 1 cm^{-1} and 10 cm^{-1} and total angular momentum of 1:

```
JTOTL  =   1,   JTOTU  =  1,   IBFIX  =   2,
EMIN   = -10.0, EMAX   = -1.0,
```

In total, this potential supports 5 vibrational states, of which the 4th and 5th are in this energy range:

```
============================================== model system: ITYPE=1 ==============================================

*****************************  ANGULAR MOMENTUM JTOT  =   1   AND SYMMETRY BLOCK  =    2  *****************************

CONVERGED ON STATE NUMBER     4 AT                              ENERGY =  -4.992074660     CM-1

CONVERGED ON STATE NUMBER     5 AT                              ENERGY =  -1.413103885     CM-1
```

3.10 Calculations in external fields

The programs can also perform calculations in external (electric, magnetic and photon) fields. This is particularly important in low-energy atomic and molecular scattering, where the scattering length may be controlled by varying external fields in the vicinity of a low-energy Feshbach resonance. The following set of (related) calculations illustrate this.

3.10.1 Using MOLSCAT to calculate the field-dependent scattering length

The following MOLSCAT calculation is for collisions of two ^{85}Rb atoms (initially in the lowest, $f = 2$, $m_f = 2$, hyperfine state), with a collision energy of 100 nK$\times k_B$, and in an external magnetic field. Feshbach resonances appear as features in the scattering length as a function of magnetic field. This calculation uses a plug-in basis-set suite, described in section 18.2. The potential used is described in chapter 14. The input file that specifies this calculation is provided as `molscat-basic_Rb2.input`.

The basis-set suite used here (`base9-alk_alk_ucpld.f`) interprets JTOTL and JTOTU as doubled values of $M_{\rm tot}$, which is the projection of the total angular momentum onto the magnetic field axis; $M_{\rm tot}$ is the only good angular momentum quantum number in the presence of a magnetic field. Its value is specified by

```
JTOTL  =   8,   JTOTU  =   8,     IBFIX  =  2,
```

Setting IBFIX = 2 specifies that only the symmetry block with $j + L + J_{\rm tot}$ even is required; for JTOT = 8, this corresponds to total parity $(-1)^{j+L} = +1$.

The collision energy is often defined with respect to the energy of the incoming atoms, which is a function of magnetic field. In this run the energy is specified as a temperature in K (EUNITS = 2) and is referred to the lowest threshold (specified by MONQN, which in this case contains doubled values of f_A, m_{fA}, f_B, m_{fB} for the required incoming channel).

```
EUNITS =   2,   NNRG   =   1,     ENERGY =  1.E-7,
                DTOL   =   1.E-6, MONQN  =  4, 4, 4, 4,
```

The scattering length is calculated at 1 G intervals over the range of interest

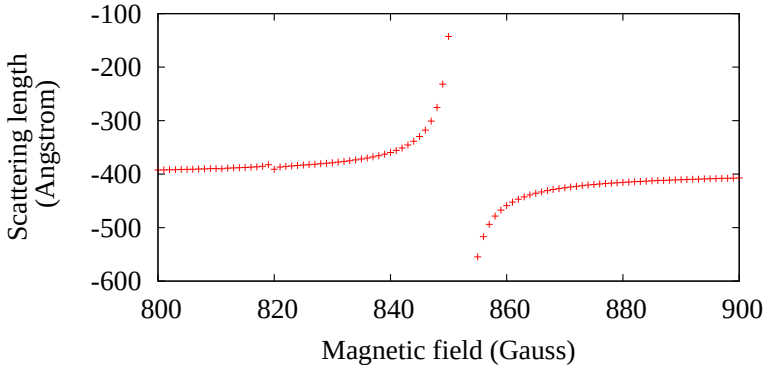
```
FLDMIN = 800.0, FLDMAX = 900.0,   DFIELD =  1.0,
```

for any channels that have low enough energy:

```
EFV SET     1:  MAGNETIC Z FIELD =   800.0000000     GAUSS
REFERENCE ENERGY IS                          -0.1780008384     CM-1    = -0.2561034905     K


THRESHOLDS CALCULATED FROM ASYMPTOTIC HAMILTONIAN:

THRESHOLD       L           ENERGY/CM-1               ENERGY/K
        1       0        -0.178000838431            -0.256103490542
[OTHER THRESHOLDS ALSO LISTED]

K-DEPENDENT SCATTERING LENGTHS/VOLUMES/HYPERVOLUMES FOR CHANNELS WITH LOW KINETIC ENERGY
CHAN   L POW     WVEC*BOHR             RE(A)/BOHR            IM(A)/BOHR
  1    0   1 2.2139874710152E-004  -3.9108818057721E+002  -5.5463354404120E-006
  2    2   4 2.2139874597323E-004  -4.0003653589240E+009  -5.0726718740952E+005
```

This output gives the scattering length for the $L = 0$ channel and the scattering hypervolume for the $L = 2$ channel. The imaginary part of the s-wave ($L = 0$) scattering length is non-zero because there is some scattering into the d-wave ($L = 2$) outgoing channel at the same threshold.

Plotting the (real part of the) s-wave scattering length as a function of magnetic field yields the following:



This plot is dominated by a wide resonance that appears as a pole near 850 G, and there is also a narrow resonance that is just visible near 820 G. The narrow resonance would need a finer grid to see well, and in fact there is also a third resonance that is too narrow to see at all with this grid.

It is not always necessary or desirable to perform calculations on a grid of fields across every resonance. MOLSCAT offers facilities for converging on resonances as a function of external field and extracting the parameters that characterise them (positions, widths, etc.). These are described in section 9.10.3.

3.10.2 Using FIELD to locate threshold crossings

Narrow resonances can be hard to locate, and may be missed entirely in calculations done on a grid which is too coarse. However, resonances occur at fields where bound states cross the energy of the colliding species. FIELD uses methods similar to BOUND to locate the fields at which bound states have a specified energy. In the absence of inelastic channels (lower in energy than the incoming channel), it is therefore possible to use FIELD to locate *all* the threshold crossings in a given range of fields. To adapt the MOLSCAT input file described above to do this, it is only necessary to remove DFIELD and change ENERGY from 100 nK. A collision energy of 0 (i.e., *at* the threshold) is usually appropriate. It is important to retain the default boundary condition BCYOMX = 0.0 to give continuity across the threshold.

The input file that specifies this calculation is provided as `field-basic_Rb2.input`. The resulting output locates the threshold crossings and contains the lines

```
CONVERGED ON STATE NUMBER  1469 AT                  MAGNETIC Z FIELD =   851.8811351     GAUSS
.
.
CONVERGED ON STATE NUMBER  1470 AT                  MAGNETIC Z FIELD =   819.5673857     GAUSS
.
.
CONVERGED ON STATE NUMBER  1471 AT                  MAGNETIC Z FIELD =   810.9519209     GAUSS
```

These include both the two resonances visible in the figure above and the third one near 810 G that is too narrow to see in the figure.

3.10.3 Using BOUND and FIELD to build the bound-state picture

It is possible to build a complete bound-state picture by running FIELD at a series of energies below threshold, or by running BOUND at a series of values of the field. In practice a combination of the two approaches is often beneficial, particularly if some bound states vary relatively fast with field and others are almost flat.

Chapter 4

Interaction types and basis sets

4.1 Interaction types

The interaction type is specified by setting ITYPE in namelist &BASIS,

$$\texttt{ITYPE} = \texttt{ITYP} + \texttt{IADD},$$

where

ITYP = 1: Linear rigid rotor + atom
ITYP = 2: Diatomic vibrotor + atom
ITYP = 3: Linear rigid rotor + linear rigid rotor
ITYP = 4: Asymmetric rigid rotor + linear rigid rotor
ITYP = 5: Symmetric top rigid rotor + atom
ITYP = 6: Asymmetric rigid rotor + atom
ITYP = 7: Diatomic vibrotor + atom, with different interaction potentials for different rotational levels
ITYP = 8: Atom + rigid corrugated surface
ITYP = 9: Plug-in code for other interaction types

and

IADD = 0: Full close-coupling (no dynamical approximation)
IADD = 10: Effective potential approximation
IADD = 20: CS (Coupled-states/centrifugal-sudden) approximation or helicity decoupling approximation
IADD = 30: Decoupled L-dominant approximation
IADD = 100: Infinite-order sudden approximation

The allowed combinations are:

		CC (+0)	EP (+10)	CS (+20)	DLD (+30)	IOS (+100)
rigid rotor + atom	1	✓	✓	✓	✓	✓
vibrating rotor + atom	2	✓	✓	✓	✓	✓
rigid rotor + rigid rotor	3	✓	✓	✓	✗	✓
asymmetric top + rigid rotor	4	✓	✗	✓	✗	✗
symmetric top + atom	5	✓	✓	✓	✗	✓
asymmetric top + atom	6	✓	✓	✓	✗	✓
vibrating rotor + atom	7	✓	✓	✓	✓	✗
atom + corrugated surface	8	✓	N/A	N/A	N/A	N/A
user-defined interaction type	9	(✓)	(✓)	(✓)	(✓)	(✓)

`ITYP` = 1 to 8 are referred to here as *built-in interaction types*.

4.2 Pair levels, pair states and pair basis functions

The programs construct sets of pair basis functions $\Phi_i(\xi_{\rm intl})$ that are used to expand the wavefunction as in Eq. 2.3. The specific set of quantum numbers needed to describe each function depends on the interaction type, as described for the built-in coupling cases in sections 4.5, 4.6, 4.7 and 4.8.

In constructing basis sets, the programs make a distinction between pair levels, pair states and pair basis functions. These terms are used somewhat differently

1. for basis sets that are diagonal in the Hamiltonian $H_{\rm intl}$ of the separated monomers and $\hat{L}^2$;
2. for basis sets that are non-diagonal in $H_{\rm intl}$ and/or $\hat{L}^2$.

The arrays used to select and specify basis sets in these two cases are described separately in the following subsections.

The programs have outer loops over variables `JTOT` and `IBLOCK`, as described in section 4.9. For the built-in interaction types, these are (mostly) used for the total angular momentum and a symmetry that is either total parity or a body-fixed projection quantum number. Before entering these loops, the programs construct restricted lists of quantum numbers that are independent of `JTOT` and `IBLOCK`; inside the loops, they construct the basis set suitable for the specific set of coupled equations that arises for that `JTOT` and `IBLOCK`.

4.2.1 Basis sets diagonal in $H_{\rm intl}$ and $\hat{L}^2$

This class includes all the built-in interaction types.

Each monomer has energy levels identified by a set of quantum numbers. A separated pair of monomers has levels identified by the product of the two sets. These are specified by a list of *pair level quantum numbers* held in the array `JLEVEL`. The corresponding set of *pair level energies* for the separated monomers are held in the array `ELEVEL`.

The quantum numbers in `JLEVEL` are limited to those needed to label degeneracy-averaged

cross sections. They do *not* include quantum numbers that have no effect on the monomer energies.

The arrays JLEVEL and ELEVEL are either input explicitly or constructed from other input data as described below. If JLEVEL is input directly or is constructed in a plug-in basis-set suite, each set of quantum numbers must be unique. The user needs an understanding of JLEVEL and ELEVEL to construct data files and interpret degeneracy-averaged cross sections output from MOLSCAT.

Before entering the loops over JTOT and IBLOCK, the pair level quantum numbers in JLEVEL are expanded internally into the (often larger) set of *pair state quantum numbers* in the array JSTATE. This includes all the quantum numbers that appear in the basis set for an interacting pair, including those that do not affect the pair energy, but *excluding* JTOT, IBLOCK and the centrifugal quantum number L (the allowed values of which depend on JTOT and IBLOCK).

In simple cases, JSTATE contains the same quantum numbers as JLEVEL. However, additional quantum numbers are sometimes needed. This is described for the built-in coupling cases in section 4.6. For example, for diatom + diatom interactions, the two monomer rotational quantum numbers j_1 and j_2 may couple to form several values of a resultant j; only pairs (j_1, j_2) are in JLEVEL, but each set (j_1, j_2, j) is stored separately in the JSTATE. The dimensions of JSTATE are (NSTATE,NQN), where NSTATE is the number of sets and NQN $-$ 1 is the number of quantum labels per set.

The programs also need access to the internal energy for each pair state. To allow this, the last element of JSTATE for each state is a pointer to the pair level energy in ELEVEL.

The user needs an understanding of the JSTATE array to interpret the program output. The quantum numbers in JSTATE are printed if IPRINT ≥ 1.

Finally, inside the loops over JTOT and IBLOCK, the programs select the *pair basis functions* that are actually used to solve each set of coupled equations. Each basis function is specified by an element of the array JSINDX (which points to a set of quantum numbers in JSTATE) and a corresponding value of L in the array L. These basis functions are then used for calculations for all the energies and external fields required for that JTOT and IBLOCK.

For basis sets diagonal in $H_{\rm intl}$ and $\hat{L}^2$, the pair basis functions are used to label the channels involved in S-matrix elements.

If IPRINT ≥ 5, the programs print a list of channels. For each channel, the list includes JSINDX, the corresponding quantum numbers in JSTATE, the value of L, the pair level index and the pair energy. In cases where L is not a good quantum number (IBOUND $= 1$, for example in helicity decoupling calculations), it is replaced by the diagonal matrix element of $\hat{L}^2$.

4.2.2 Basis sets non-diagonal in $H_{\rm intl}$ or $\hat{L}^2$

For the built-in interaction types, the basis functions are eigenfunctions of the Hamiltonian $H_{\rm intl}$ of the separated monomers. However, for plug-in basis-set suites this is not essential: there may be off-diagonal (but R-independent) terms in $H_{\rm intl}$ and/or $\hat{L}^2$ that are programmed as described in section 17.9.

Under these circumstances, before entering the loops over JTOT and IBLOCK, a plug-in suite constructs an array JSTATE that contains values of all quantum numbers that appear in the basis set for an interacting pair except JTOT, IBLOCK and L. Some plug-in suites also construct the array JLEVEL, but it is not used outside the plug-in suite so it has no particular significance.

Inside the loops over JTOT and IBLOCK, the plug-in suite constructs a *primitive basis set* using the arrays JSINDX and L in the same way as described in section 4.2.1.

If IPRINT ≥ 5, the programs print a list of the primitive basis functions. For each basis function, the list gives JSINDX, the corresponding quantum numbers in JSTATE and the value of L. In cases where the basis set is diagonal in L but L does not have an integer value (NRSQ $= 0$ and IBOUND $= 1$), it is replaced by the diagonal matrix element of $\hat{L}^2$.

The primitive basis set is used to construct the coupled equations. However, a single basis function does *not* correspond to an energy level of a pair of separated monomers and cannot be used to label S-matrix elements. Under these circumstances, the programs diagonalise $H_{\rm intl}$ for each JTOT and IBLOCK (and each external field) to find its eigenvalues and eigenvectors. The eigenvalues are the channel threshold energies, and are available for use as reference energies in all three programs. They are printed if IPRINT ≥ 6 for MOLSCAT and BOUND, or 10 for FIELD (because for FIELD they change as a function of field during the course of locating each bound state). The corresponding eigenvectors are printed if IPRINT ≥ 15.

BOUND and FIELD calculate the log-derivative matching matrix (section 2.7) in the primitive basis set. They use the channel threshold energies only for reference energies. However, MOLSCAT transforms the wavefunction matrix $\mathbf{\Psi}(R_{\rm max})$ (or the log-derivative matrix $\mathbf{Y}(R_{\rm max})$) into the *asymptotic basis set* that diagonalises $H_{\rm intl}$ and $\hat{L}^2$ (and, optionally, extra operators to resolve degeneracies in the eigenvalues of $H_{\rm intl}$). Scattering boundary conditions are then applied in the asymptotic basis set.

4.3 Convergence of the basis set

Calculated scattering and properties and bound-state positions depend on the size of the basis set. For basis sets off-diagonal in $H_{\rm intl}$, even the energies of the separated monomers may depend on the size of the basis set.

The interaction potential and off-diagonal terms in $H_{\rm intl}$ and/or $\hat{L}^2$ couple basis functions arising from different monomer levels. It is always important to establish that the basis set used is large enough to give the desired accuracy. The size of basis set required may often be estimated by physical intuition, sometimes assisted by perturbation theory, but there is usually no substitute for carrying out tests with increasingly large basis sets and checking the convergence of the properties of interest.

It is *not* usually adequate to carry out scattering calculations that include only asymptotically open channels. Closed channels can have substantial effects. Bound-state calculations are often even more sensitive to the inclusion of high-lying basis functions than scattering calculations.

4.4 Units of energy for quantities in &BASIS

The units of energy for quantities input in &BASIS are *independent* of those used for quantities in &INPUT. They are specified by EUNITS, which is an integer that selects a unit of energy from the list in section 3.7. The default for EUNITS is 1, indicating energies expressed as wavenumbers in cm^{-1}.

If the energy unit required is not among those listed, EUNITS may be set to 0 and the required value (in units of cm^{-1}) supplied in EUNIT. If this is done, the name of the unit should be supplied in the character variable EUNAME.

EUNITS, EUNIT and EUNAME are distinct from the variables with the same names in namelist &INPUT (section 6.1), but their allowed values and interpretation are the same.

4.5 Built-in interaction types: pair level quantum numbers and energies

The user *may* specify a list of pair level quantum numbers in the array JLEVEL and a corresponding list of energies in the array ELEVEL. Alternatively (and more commonly), JLEVEL and/or ELEVEL are generated from other input quantities:

NLEVEL: if > 0, NLEVEL indicates that the quantum numbers of the levels to be used in constructing the basis set are input as NLEVEL sets of values in the array JLEVEL. If NLEVEL $= 0$, the quantum numbers for the levels are calculated internally as described below.

JLEVEL: integer array specifying the pair level quantum numbers. The array JLEVEL is structured differently for each value of ITYP as described below. If NLEVEL > 0, JLEVEL must be supplied explicitly in the input file. If NLEVEL $= 0$, the pair level quantum numbers are calculated internally from input quantities that specify ranges and step sizes for them, as described for each value of ITYP below. JLEVEL is declared as a one-dimensional array (current dimension 4000, set in module sizes), although it is conceptually two-dimensional for ITYP > 1.

ELEVEL: array of NLEVEL pair level energies, corresponding to the pair levels in the array JLEVEL (current dimension 1000, set in module sizes). If all the elements of ELEVEL are 0.0, or NLEVEL is 0, the energies are calculated from input values of monomer spectroscopic constants as described for each value of ITYP below.

The methods of specifying the pair levels to be included are independent of any dynamical approximations employed, so that the information given for each ITYP below is applicable to ITYPE $=$ ITYP, ITYP $+ 10$, ITYP $+ 20$ and ITYP $+ 30$. For IOS cases (ITYPE $=$ ITYP $+ 100$), the required input is generally the same, except that rotational energies are not required for IOS calculations.

4.5.1 Linear rigid rotor + atom (ITYP = 1)

The basis set used for a linear rigid rotor is formed from spherical harmonics Y_j^m. These may be functions of either spaced-fixed angles (β, α) (for close-coupling calculations) or body-fixed angles (θ, ϕ) (for coupled-states and helicity-decoupling calculations).

The array JLEVEL must contain a list of values of j for monomer rotational states. It is usually generated from input parameters JMIN, JMAX and JSTEP, which have the obvious meanings. For special purposes, NLEVEL may be set greater than zero and a list of NLEVEL j values supplied in the array JLEVEL.

The energy levels are usually calculated from

$$E(j) = (B_e - \alpha_e/2)j(j+1) - D_e[j(j+1)]^2. \tag{4.1}$$

The input parameters BE, ALPHAE and DE have the obvious meanings. Since the rotational constant actually used is simply BE $- 0.5 \times$ ALPHAE, this value may be input directly in BE with ALPHAE omitted from the namelist if preferred.

4.5.2 Diatomic vibrotor + atom (ITYP = 2 and 7)

The monomer basis set for a vibrating diatom is formed from products of spherical harmonics and vibrational wavefunctions with quantum number v in the monomer internuclear separation. Once again the spherical harmonics may be functions of either spaced-fixed angles (β, α) (for close-coupling calculations) or body-fixed angles (θ, ϕ) (for coupled-states and helicity-decoupling calculations).

NLEVEL must be set greater than zero and the pair levels to be included must be specified as a list of NLEVEL (j, v) pairs supplied in the array JLEVEL (in the order $j_1, v_1, j_2, v_2, \ldots j_{\texttt{NLEVEL}}, v_{\texttt{NLEVEL}}$). There is no option for generating this list from limits on quantum numbers.

The energy levels may be specified as a list in the array ELEVEL. Alternatively, if all NLEVEL elements of ELEVEL are zero (the default), the energy levels are calculated from

$$E(j,v) = \omega_e v - \omega_e x_e v(v+1) + \left[B_e - \left(v + \tfrac{1}{2}\right)\alpha_e\right] j(j+1) - D_e[j(j+1)]^2.$$

The input parameters WE, WEXE, BE, ALPHAE and DE have the obvious meanings. Note that the energies are defined with respect to $E(0,0)$ even if the $(0,0)$ pair is not included in JLEVEL.

For IOS calculations the vibrational quantum numbers are selected from the JLEVEL array, and any rotational quantum numbers supplied are ignored, except that the maximum value is used to limit which cross sections are calculated. If vibrational energies are supplied in ELEVEL, the first one encountered for each vibrational manifold is kept. If all ELEVEL are zero, energies are generated from WE and WEXE. If the input NLEVEL is negative, a user-supplied routine GET102 must be provided to set NLEVEL, JLEVEL and ELEVEL values. The distribution includes a dummy version of this routine.

4.5.3 Linear rigid rotor + linear rigid rotor (ITYP = 3)

The basis set is formed from coupled products of spherical harmonics $Y_{j_1}^{m_1}$ and $Y_{j_2}^{m_2}$.

The array JLEVEL must contain a list of pairs of rotational quantum numbers (j_1, j_2). It is usually generated from input parameters J1MIN, J1MAX, J1STEP for molecule 1 and J2MIN, J2MAX, J2STEP for molecule 2. For special purposes, NLEVEL may be set greater than zero and a list of NLEVEL (j_1, j_2) pairs supplied in the array JLEVEL (in the order j_{11}, j_{21}; j_{12}, j_{22}; ...; $j_{1,\mathrm{NLEVEL}}, j_{2,\mathrm{NLEVEL}}$).

The energy levels are usually specified as a sum of two terms of the form

$$E(j) = (B_e - \alpha_e/2)j(j+1) - D_e[j(j+1)]^2. \tag{4.2}$$

The input parameters BE(1,2) ALPHAE(1,2) and DE(1,2) specify the parameters for molecules 1 and 2. If IDENT = 1 and the values for molecule 2 are zero, the program sets them equal to the values for molecule 1. Since the rotational constants actually used are simply BE $- 0.5 \times$ ALPHAE, these values may be input directly in BE(1,2) with ALPHAE omitted from the namelist if preferred.

If the two molecules are identical, the basis functions corresponding to (j_1, j_2) and (j_2, j_1) are indistinguishable. In this case, the input variable IDENT (default 0) should be set to 1 and (if JLEVEL is supplied as a list) only distinguishable pairs (i.e., $j_1 \geq j_2$) should be included. The programs then carry out separate calculations for states with odd and even exchange symmetry, in separate symmetry blocks.[1]

For identical molecules, it is also necessary to specify the statistical weights to be applied to the different symmetry combinations when calculating cross sections. The statistical weights for antisymmetric and symmetric combinations of (j_1, j_2) and (j_2, j_1) may be specified explicitly in the WT array as WT(1) and WT(2) respectively. If both WT(1) and WT(2) are zero (the default), they are calculated from the single nuclear spin input in SPNUC: for integer SPNUC (bosonic particles), WT(1) = SPNUC/(2 * SPNUC + 1) and WT(2) = (SPNUC + 1)/(2 * SPNUC + 1). For half-integer SPNUC (fermionic particles), the two values are exchanged.

For identical molecules, all the programs skip any symmetry block for which WT is zero.

There are different definitions in the literature for state-to-state integral cross sections for two identical particles that are in the same state either before or after the collision. The issues have been discussed by Huo and Green [46]. Versions of MOLSCAT before 2020.0 evaluated such cross sections for ITYP = 3 using the expression of Takayanagi, Eq. (2.18) of ref. [46]. However, version 2020.0 uses Eq. (2.16) of ref. [46]; this contains no additional factors of 2 when the initial or final states are the same, and gives the same values for cross sections $\sigma_{jj \to j'j'}$ when calculated with or without identical-particle symmetry.

[1] In BOUND and FIELD, a calculation that neglects identical-particle symmetry when it is present simply combines the coupled equations for odd and even exchange symmetry. It is less efficient, but gives the same eigenvalues and wavefunctions, provided both sets of symmetry-related functions are included. In MOLSCAT, the S-matrix elements obtained in the symmetrised and unsymmetrised cases are related by simple summations and factors of $\sqrt{2}$, but some elements that are off-diagonal in the unsymmetrised case are transferred to the diagonal in the symmetrised case. Because of the different expressions used for elastic (diagonal) and inelastic (off-diagonal) cross sections (due to the presence of δ_{if} in Eq. 2.12), there are no simple relationships between cross sections connecting symmetry-related pairs in the two cases.

4.5.4 Rigid symmetric (or near-symmetric) top + atom (ITYP = 5)

The basis set used for a rigid symmetric top is formed from Wigner rotation matrices D^{j*}_{mk}, where k is the projection of the angular momentum j onto the symmetry (z) axis of the top. The rotation matrices may be functions of either spaced-fixed Euler angles (α, β, γ) (for close-coupling calculations) or body-fixed angles (ϕ, θ, χ) (for coupled-states and helicity-decoupling calculations).

The array JLEVEL must contain a list of sets of 3 rotational quantum numbers $(j, k, \mathtt{PRTY})$. PRTY indicates that the function is proportional to an even or odd linear combination of rotation matrices,

$$D^{j*}_{m\,k} + (-1)^{\mathtt{PRTY}} D^{j*}_{m\,-k}, \tag{4.3}$$

normalised as appropriate. For $k = 0$, only the even combination is possible. The monomer parity is $(-1)^{j+\mathtt{PRTY}}$.

The array JLEVEL is usually generated from input parameters JMIN, JMAX and JSTEP. If JSTEP = 2, only functions of the same parity as j = JMIN, $k = 0$ are included; note that this produces one function (not 2) for each (j, k) even when $k > 0$.

The additional input variable KMAX (default 0, equivalenced to KSET) may be used to limit the k values included: if it is negative, only levels with $k = |\mathtt{KSET}|$ are included; if it is zero or positive, only levels with $k \leq$ KMAX are included. If *all* k levels are required, KMAX should be set to at least JMAX (999 recommended).

For special purposes, NLEVEL may be set greater than zero and a list of NLEVEL triples $(j, k, \mathtt{PRTY})$ supplied in the array JLEVEL.

If the energy levels are not given explicitly in ELEVEL, they are calculated from the standard near-symmetric top equation using rotational constants supplied in the input variables A, B and C. Neglecting centrifugal distortion, l-type doubling and tunnelling, the expression is

$$E(j,k) = \frac{(A+B)}{2}\left[j(j+1) - k^2\right] + Ck^2. \tag{4.4}$$

Note that A, B and C *must* correspond, respectively, to moments of inertia about the x, y, and z axes used in the description of the interaction potential (see below) *not* necessarily in descending order of magnitude, as is usual in spectroscopy. The program requires that the z axis be the symmetry axis of the top (or the axis of near symmetry) and that the xz plane be a reflection plane. For a symmetric top, then, A = B and C is the unique constant.

3-fold symmetry

The most common use of ITYP = 5 is for a symmetric top like ammonia, with 3-fold permutation symmetry, and inversion doubling. The code supports this, allowing the user to set ISYM(3) to restrict k to values that satisfy either $3n \pm 1$ (ISYM(3) = 1) or $3n$ (ISYM(3) = 3), and ISYM(4) to indicate the nature of the 3 identical atoms (0 for bosons and 1 for fermions). In addition, if ROTI(10) is set non-zero then it is used as the zeroth-order tunnelling splitting ν_0. ROTI(11) and ROTI(12) are used for the centrifugal distortion components of the tunnelling splitting, ν_a and ν_b,

$$\nu = \nu_0 - \nu_a\left[j(j+1) - k^2\right] - \nu_b k^2. \tag{4.5}$$

Linear molecules with orbital or vibrational angular momentum

ITYP = 5 can also be used to handle interactions between atoms and linear molecules with either electronic orbital angular momentum Λ (electronic state Π, Δ, etc.) or vibrational angular momentum l. The rotational functions for such molecules are formed from rotation matrices $D^{j*}_{m\Lambda}$ or D^{j*}_{ml}. In this case the basis set usually contains only a single value of Λ or l: this may be selected by setting KSET to a negative value, KSET $= -|\Lambda|$ or $-|l|$. This option does not handle electron spin.

For molecules with $D_{\infty h}$ symmetry (homonuclear diatomics, or molecules such as CO_2 or HCCH), JSTEP should be set to 2 to include only functions of the same parity as $j =$ JMIN, $k = 0$ as above.

4.5.5 Asymmetric (or spherical) top + atom (ITYP = 6)

Asymmetric top wavefunctions are formed from linear combinations of Wigner rotation matrices,

$$|j,\tau,m\rangle = \left(\frac{2j+1}{8\pi^2}\right)^{\frac{1}{2}} \sum_k a^j_{\tau,k} D^{j*}_{mk}. \tag{4.6}$$

The rotation matrices may be functions of either spaced-fixed Euler angles (α,β,γ) (for close-coupling calculations) or body-fixed angles (ϕ,θ,χ) (for coupled-states and helicity-decoupling calculations).

The implementation of ITYP = 6 requires that the molecular xz plane is a plane of symmetry. It is not general enough to handle chiral asymmetric tops.

The array JLEVEL must contain a list of pairs (j,τ), where τ is an index for the level, $1 \le \tau \le 2j+1$. JLEVEL is usually generated from input parameters JMIN, JMAX and JSTEP as described below.

The monomer energy levels $E_{j\tau}$ and wavefunction coefficients $a^j_{\tau,k}$ are usually calculated internally from rotational constants. If all three rotational constants (A, B and C) are specified, the program constructs and diagonalises the asymmetric top Hamiltonian, which (neglecting centrifugal distortion) is

$$\hat{H}_{\rm rot} = Aj_a^2 + Bj_b^2 + Cj_c^2. \tag{4.7}$$

Note that (as for ITYP = 5) A, B and C *must* correspond respectively to the x, y and z axes used in the potential expansion. They are *not* necessarily in descending order of magnitude, as is usual in spectroscopy.

Centrifugal distortion constants D_J, D_{JK} and D_K may also be supplied in the input variables DJ, DJK, and DK, respectively, and contribute an energy term

$$-D_J j(j+1) - D_{JK} j(j+1)k^2 - D_K k^4. \tag{4.8}$$

However, these do not correspond to the constants conventionally used for asymmetric tops.

Rotational levels are calculated for j from JMIN to JMAX in steps of JSTEP. If EMAX > 0, a level is kept only if it has energy below EMAX. A level is also kept only the corresponding wavefunction coefficients $a^j_{\tau,k}$ meet the symmetry restrictions imposed by ISYM, which is described below.

If IASYMU < 0, the resulting wavefunctions are written on unit |IASYMU| in the format described below for input from IASYMU. The energies written to IASYMU are in the units specified by EUNIT or EUNITS, so that subsequent calculations must use the same units.

Spherical tops fit into the same framework as asymmetric tops, but with a different rotational Hamiltonian governed by a single rotational constant and a tetrahedral centrifugal distortion constant d_t that splits rotational levels with $j > 1$ into sets of A_1, A_2, E, T_1 or T_2 symmetry (where the last two are commonly called F_1 and F_2 in the spectroscopic literature). To select this option, set A = B = C and input a non-zero value d_t in DT. Note that d_t must have magnitude greater than about 10^{-7} cm^{-1}: if it is set to zero, the rotational states are not resolved into their contributing symmetries.

If rotational constants are not input, the program reads the arrays JLEVEL and ELEVEL from unit IASYMU, together with the corresponding wavefunction coefficients $a^j_{\tau,k}$. Each level is described on IASYMU with the triple $j, \tau, E_{j\tau}$ where $E_{j\tau}$ is the rotor energy in &BASIS energy units. For each level, $2j + 1$ wavefunction coefficients $a^j_{\tau,k}$ are required (corresponding to $k = -j, -j + 1, ..., j$); these must follow the $j, \tau, E_{j\tau}$ card with format (6F12.8) on $(n_k + 5)/6$ subsequent cards. Coefficients need not be normalised, but they are checked for valid symmetries.

If IASYMU $= 5$ (standard input), data records should follow namelist &BASIS and precede namelist &POTL; in this case a positive value of NLEVEL must be set in &BASIS, to specify the number of rotor functions given in the file.

If IASYMU is an auxiliary unit, it is possible to read levels until an end-of-file condition occurs; specifying NLEVEL < 0 selects this option. If NLEVEL $= 0$, input functions are skipped if j is not in the range JMIN to JMAX in steps of JSTEP (if JMAX > 0) or if $E_{j\tau} >$ EMAX (if EMAX > 0). It is possible to specify a subset of the levels on IASYMU by setting NLEVEL to a negative value. In that case, input functions are skipped unless j_1 and τ match values in JLEVEL$(2i-1)$, JLEVEL$(2i), i = 1, |$NLEVEL$|$.

Note that for IOS calculations (ITYPE $= 106$) rotor wavefunctions are used only to calculate state-to-state cross sections from the "generalised IOS" cross sections. For this case generation of rotational wavefunctions from rotational constants is not implemented and they must be explicitly supplied as data on IASYMU if state-to-state cross sections are required.

Symmetries of asymmetric top and spherical top functions (also relevant to ITYPE $= 4$)

Asymmetric top functions transform according to the point group D_2 and are characterised by two symmetries:

1. the k values involved are *either* even *or* odd (corresponding to $+$ or $-$ symmetry with respect to a C_2 rotation about the z axis)
2. the functions are of the form $\sum_k c_k(D^{j*}_{mk} + \epsilon D^{j*}_{m,-k})$ where ϵ is either $+1$ or -1.

Internally, the program assigns one of four symmetry types to each asymmetric top function:

PRTY	even/odd k	ϵ
0	even	$+1$
1	even	-1
2	odd	$+1$
3	odd	-1

Asymmetric top functions input from IASYMU must conform to these symmetries.

The lists of quantum numbers printed for asymmetric and spherical tops include j, τ, PRTY and an index that identifies the location where the corresponding eigenvector is stored internally. The index has no physical significance.

If the monomer functions are calculated from rotational constants (but not if they are input explicitly from IASYMU), the symmetry types that are actually included in the basis set may be restricted using the input variable ISYM(1). This is interpreted bitwise, and a particular asymmetric rotor function is included if it passes *all* the specified tests.

If bit 0 is set,	odd k functions are excluded	add 1 to ISYM
If bit 1 is set,	even k functions are excluded	add 2 to ISYM
If bit 2 is set,	functions with $(-1)^j\epsilon = -1$ are excluded	add 4 to ISYM
If bit 3 is set,	functions with $(-1)^j\epsilon = +1$ are excluded	add 8 to ISYM
If bit 4 is set,	functions with degeneracy 1 are excluded	add 16 to ISYM
If bit 5 is set,	functions with degeneracy 2 are excluded	add 32 to ISYM
If bit 6 is set,	functions with degeneracy 3 are excluded	add 64 to ISYM
If bit 7 is set,	functions with degeneracy > 3 are excluded	add 128 to ISYM

To construct ISYM(1), start with 0 (including all functions) and add 2^n for each class of functions to be excluded.

To decide which states are coupled, consider the terms (λ, κ) present in the expansion of the interaction potential:

1. If only terms with κ even are present, functions with k even are not coupled to those with k odd.
2. If only terms with $(\lambda + \kappa)$ even are present, functions with $(-1)^j\epsilon$ even are not coupled to those with $(-1)^j\epsilon$ odd.

If either or both of these symmetries is present, it is most efficient to do separate calculations for the (two or four) different sets.

For the special cases of $J_{\rm tot} = 0$ and coupled states with $K = 0$ (symmetry block IBLOCK $= 1$), there are additional restrictions on the functions that are coupled: $j + j' + \lambda$ must be even, and ϵ must be conserved. Do not be misled into believing that these symmetry restrictions hold in more general cases.

The flags that test degeneracy are intended for use with spherical tops, and allow A, E and F (T) levels to be included selectively. For E levels, both functions in the degenerate pair are needed and may be selected with ISYM $= 208$ (128+64+16). For F (T) levels, only a single function (with even k) is needed from each degenerate set, and may be selected with ISYM $= 177$ (128+32+16+1).

For asymmetric rotors, functions for different values of τ are non-degenerate. However, near-

degeneracies can occur, and the program interprets these as degeneracies if the levels (from diagonalisation) are within a tolerance (currently 10^{-9}, set as EPS in SET6). To be safe, do not set any high-bit flags for asymmetric rotors.

The levels to be included may also be restricted using the variable EMAX if desired.

4.5.6 Asymmetric rigid rotor + linear rotor (ITYP = 4)

The basis functions here are combinations of asymmetric top rotor wavefunctions and linear rotor wavefunctions [23]. The array JLEVEL must contain a list of sets of 3 rotational quantum numbers (j_1, τ, j_2).

Input of the asymmetric top functions follows the capabilities for ITYP = 6, as described above. If rotational constants (A(1), B(1), and C(1)) are specified, asymmetric top wavefunctions are calculated, subject to limits specified by JMIN, JMAX, JSTEP (with synonyms J1MIN, J1MAX, and J1STEP), EMAX, and ISYM. Otherwise, wavefunctions are read from unit IASYMU, as for ITYP = 6.

The asymmetric top wavefunctions are combined with linear rotor functions specified by J2MIN, J2MAX, J2STEP; energies of the linear rotor are obtained from the rotational constant, which must be supplied as BE(2). If specified, ALPHAE(2) and DE(2) are also used as for ITYP = 1. If EMAX > 0 is specified, it is used to limit the pair levels to those with energies (asymmetric top plus linear rotor) less than EMAX.

If NLEVEL < 0, the programs use |NLEVEL| sets of quantum numbers from the array JLEVEL and read asymmetric top energy levels and wavefunction coefficients from unit IASYMU. The asymmetric top functions for all required j_1, τ must be available on unit IASYMU. In this case it is also possible to specify the energies (asymmetric top plus linear rotor) in the ELEVEL array; otherwise energies are taken from IASYMU for the asymmetric top and computed from BE(2) in the same way as for ITYP = 1 for the linear rotor. The format of unit IASYMU is described in section 4.5.5.

4.5.7 Atom + rigid corrugated surface (ITYP = 8)

MOLSCAT and BOUND can calculate S matrices or bound states respectively for diffractive scattering of atoms from a rigid corrugated (solid) surface [28, 29] using ITYPE = 8. In this case the basis functions depend on the energy and angles of incidence, which are specified in namelist &INPUT, so both &INPUT and &BASIS are described here.

For ITYP = 8 the reduced mass is the same as the atomic mass m and is input in URED.

Surface scattering calculations do not require loops over total angular momentum and symmetry block. The programs therefore use the internal loop over JTOT to loop over the polar angle θ (measured from the surface normal) and the loop over symmetry block IBLOCK to loop over azimuthal angle ϕ (measured relative to the surface reciprocal lattice vector g_1). The incident wavevector $\boldsymbol{k}$ may be decomposed into components $k_\perp = k\cos\theta$, perpendicular to the surface, and a vector $\boldsymbol{K}$ in the surface plane, with magnitude $k_\parallel = k\sin\theta$. The loop over θ is controlled by the input parameters JTOTL, JTOTU, JSTEP, THETLW and THETST, while that

over ϕ is controlled by MXPHI, PHILW and PHIST. The logic used is equivalent to:

```
DO JTOT = JTOTL, JTOTU, JSTEP
  THETA = THETLW + THETST*JTOT
  DO M = 1, MXPHI
    PHI = PHILW + PHIST*(M-1)
          ..
          ..
```

Scattering calculation for parallel momentum corresponding to angles THETA, PHI and incident energy ENERGY(1), or

bound-state calculations for parallel momentum corresponding to angles THETA, PHI and wavevector 100 Å^{-1}.

```
          ..
          ..
  ENDDO
ENDDO
```

The basis functions for ITYPE = 8 are formed from surface reciprocal-lattice vectors $\boldsymbol{G} = (g_1, g_2)$, and are proportional to

$$\exp[\mathrm{i}(\boldsymbol{K} + \boldsymbol{G}) \cdot \boldsymbol{R}], \tag{4.9}$$

where $\boldsymbol{R}$ is the position of the atom within the surface unit cell. The dimensions and symmetry of the surface unit cell are specified in the input array ROTI. ROTI(1) and ROTI(2) are the lengths of the real-space lattice vectors (in Å, irrespective of the units of length used elsewhere). ROTI(3) is the lattice angle in degrees.

Note that, for surface scattering, subsequent energies have a rather non-intuitive meaning, because the "energy" that appears in the coupled equations is $\hbar^2 k_\perp^2/2m$, while the parallel component $\boldsymbol{K}$ of the momentum enters in the threshold energies $\hbar^2|\boldsymbol{K} + \boldsymbol{G}|^2/2m$. The programs interpret subsequent energies as having the same parallel momentum as the first energy, but a different perpendicular momentum. In BOUND, this allows calculations of energies as a function of $\boldsymbol{K}$ (a band-structure diagram). In MOLSCAT however, it corresponds to a change in the polar angle as well as the scattering energy, and the program calculates and prints the new polar angle.

The same loops are used by BOUND to specify the parallel component of the momentum, so the bound states located are for the same $k_\perp$ as at EMIN, rather than at the same polar angle. The same is in principle true for FIELD, but it is hard to think of a use for ITYPE = 8 with FIELD.

Basis sets for surface scattering are generated in two steps. At the time that namelist &BASIS is read, the program sets up a master list of basis functions that *may* be included in subsequent scattering calculations. This takes place before entering the loops over incident angles and energies, so must be angle- and energy-independent. If NLEVEL = 0, the list of basis functions

is generated from J1MAX, J2MAX; g_1 loops from $-$J1MAX to $+$J1MAX, and g_2 loops from $-$J2MAX to $+$J2MAX. However, each basis function $\boldsymbol{G} = (g_1, g_2)$ is included only if the parallel kinetic energy $\hbar^2|\boldsymbol{G}|^2/2m <$ EMAX.

Alternatively, if NLEVEL > 0, the JLEVEL array must contain a list of (g_1, g_2) pairs. The test involving EMAX is bypassed in this case.

Subsequently, for each value of incident θ, ϕ and energy, the program calculates the parallel component $\boldsymbol{K}$ of the incident momentum, and selects from the master list those basis functions for which $\hbar^2|\boldsymbol{K}+\boldsymbol{G}|^2/2m <$ EMAXK. This occurs even if the JLEVEL array is specified explicitly in namelist &BASIS.

Thus, only those basis functions that satisfy both the EMAX and EMAXK criteria are ultimately included in the basis set. Since the EMAXK criterion is usually the most sensible physically, EMAX serves principally to keep the automatically generated master list within manageable bounds; the master list must not contain more than MXELVL functions (current value 1000, set in module sizes).

For the common case of surface scattering with the incident beam approaching along a symmetry direction, the programs automatically construct the appropriate symmetrised linear combinations of basis functions. This takes place as part of the second step described above, since it is not until that stage that the program knows the incident angle.

4.6 Additional quantum numbers in JSTATE but not JLEVEL

The array of pair state quantum numbers JSTATE always includes all the pair level quantum numbers in JLEVEL and an entry (the last one for each pair state) that identifies the state-to-state cross sections to which it contributes for basis sets diagonal in $H_{\rm intl}$ and $\hat{L}^2$. This entry is a pointer to the JLEVEL and ELEVEL arrays. For asymptotically non-diagonal basis sets, this entry is not used, and instead the asymptotic energies are stored in the ELEVEL array as they are calculated, and an array INDLEV is used to index them.

For ITYP $= 1$, 2, 5 and 7, the quantum numbers in JSTATE are the same as those in JLEVEL.

For ITYP $= 3$, both species have angular momentum, j_1 and j_2. The basis sets used couple these together to form a resultant j, which can take values from $|j_1 - j_2|$ to $j_1 + j_2$. JSTATE has an additional entry for j and lists each resulting pair state separately, so NQN $= 4$. The exception to this is the effective potential approximation (ITYPE $= 13$), where there is no j quantum number and NQN $= 3$.

For ITYP $= 6$, JSTATE includes an entry for PRTY, which describes the symmetry of the rotor function as discussed in section 4.5.5. In addition, JSTATE contains $i_{j\tau}$ and $n_{j\tau} = 2j + 1$, resulting in NQN $= 6$; $i_{j\tau}$ is a pointer to the first of $n_{j\tau}$ wavefunction coefficients $a^j_{\tau,k}$ in the array ATAU. In the present implementation, the array ATAU is allocated storage at the top of the array JSTATE, above the sets of integers.

For ITYP $= 4$, JSTATE includes all the additional entries described for ITYP $= 6$ and one for the resultant j of j_1 and j_2, so NQN $= 8$.

4.7 Close-coupling calculations

The programs implement close-coupling calculations (calculations without dynamical approximations) in a space-fixed basis set for ITYPE = 1 to 7. The full space-fixed basis functions are formed by coupling j to the end-over-end quantum number L to form the total angular momentum $J_{\rm tot}$

$$|(j,L)J_{\rm tot}M_{\rm tot}\rangle = \sum_{m_j,M_L} \langle jm_j, LM_L|J_{\rm tot}M_{\rm tot}\rangle |jm_j\rangle|LM_L\rangle. \tag{4.10}$$

The resulting coupled equations are independent of $M_{\rm tot}$. The calculations are carried out for one value of $J_{\rm tot}$ and symmetry block IBLOCK at at time. For close-coupling calculations, IBLOCK encodes the total parity $(-1)^{j+L}$, which is $(-1)^{J_{\rm tot}+\texttt{IBLOCK}}$. For ITYPE = 3, IBLOCK also encodes identical particle symmetry if present, with the 2 blocks for odd exchange symmetry followed by the 2 blocks for even exchange symmetry.

The full basis set includes all combinations of L with functions specified by JSTATE that have the required $J_{\rm tot}$ and total parity. Inside the loops over JTOT and IBLOCK, the programs construct arrays JSINDX and L, of dimension N. For each basis function i, JSINDX(i) is a pointer to quantum numbers in the JSTATE array and L(i) is the corresponding value of L. For specific values of $J_{\rm tot}$ and parity, some functions in JSTATE may not appear at all, and others may appear multiple times in combination with different values of L.

For special purposes, basis functions for high L may be excluded by setting ISYM2(1) to the upper limit required for L(i).

4.8 Decoupling approximations

4.8.1 Effective potential

The effective potential method of Rabitz [30] is supplied and coded as IADD = 10, but it has not been used for many years and is no longer supported. It uses 'effective rotational states', which are nondegenerate and do not couple to the orbital angular momentum L, so that L and $J_{\rm tot}$ are identical.

4.8.2 CS (Coupled-states / centrifugal sudden) and helicity decoupling

The monomer basis functions can be expressed relative to a rotating, body-fixed coordinate system, where now the projection quantum numbers m (designated K in this case) refer to projection on the interparticle axis rather than the space-fixed Z axis. The resulting body-fixed basis set is related to the space-fixed set by a unitary transformation, but now the interaction matrix is diagonal in K. However, the operator $\hat{L}^2$ is non-diagonal in K.

There are several different approximations that neglect the matrix elements of $\hat{L}^2$ which are off-diagonal in K. All these are implemented with IADD = 20, with symmetry block IBLOCK (see below) being used for coupled equations with K = IBLOCK − 1. Since $-K$ is equivalent to $+K$, only $K \geq 0$ is implemented.

The helicity decoupling approximation neglects matrix elements off-diagonal in K but evaluates the diagonal matrix elements exactly as

$$\langle jKJ_{\rm tot}|\hat{L}^2|jKJ_{\rm tot}\rangle = J_{\rm tot}(J_{\rm tot}+1) + j(j+1) - 2K^2 \tag{4.11}$$

This is invoked with IBOUND $= 1$. In this approach JTOT is interpreted as $J_{\rm tot}$, so $K \leq J_{\rm tot}$ and only basis functions with $j \geq K$ are included in the basis set.

The helicity decoupling approximation is very useful for bound states of some Van der Waals complexes. However, the diagonal matrix elements of $\hat{L}^2$ do not correspond to values that can be expressed as $L(L+1)$ with integer L. MOLSCAT applies boundary conditions using Ricatti-Bessel functions of non-integer order that properly take account of the diagonal centrifugal potentials.[2]

The L-labelled coupled-states approximation of McGuire and Kouri [31] is also implemented with IADD $= 20$, but requires IBOUND $= 0$. This makes the *centrifugal sudden* approximation in which L is set to the same value for all basis functions in each set of coupled equations. The sums over $J_{\rm tot}$ that appear in cross sections (e.g., Eq. 2.12) are replaced by sums over L, and JTOT is interpreted as L not $J_{\rm tot}$. In this case, although the basis set is still restricted to functions $j \geq K$, K is *not* restricted by $K \leq$ JTOT.

By default, CS and DLD calculations (see below) are executed for all values of the body-fixed projection number K up to the largest value of j in the basis set. However, if cross sections are required between only the lowest few levels, this is unnecessary. If JZCSMX is set > -1 on input, K is limited by JZCSMX instead of JMAX. Cross sections involving levels with $j >$ JZCSMX are then not valid.

In CS calculations, JZCSFL (default 0, allowed values $-1, 0, 1$) sets the orbital quantum number in each channel I to L(i) = IABS(JTOT + JZCSFL$*j(i)$). This is a historical remnant and values other than the default are not recommended.

4.8.3 Decoupled L-Dominant

This decoupling approximation [33] is supplied and coded as IADD $= 30$, but it has not been used for many years and is no longer supported.

JZCSMX may be used with DLD calculations, as described for CS calculations above.

4.9 Loops over JTOT and IBLOCK

The loop over JTOT is controlled by the three input variables JTOTL, JTOTU and JSTEP; the program loops from JTOTL to JTOTU in steps of JSTEP.

The loop over symmetry blocks IBLOCK is used for different purposes for different interaction types. By default, the programs loop over all possible values of IBLOCK for the interaction type concerned. However, if the input variable IBFIX is non-zero and IBHI $<$ IBFIX, the

[2] In versions of MOLSCAT before 2014, the integer values contained in L were used in the scattering boundary conditions.

programs perform calculations only for IBLOCK = IBFIX. If IBHI ≥ IBFIX, calculations are performed for the range of IBLOCK values from IBFIX to IBHI.

It should be noted that bound-state energies can shift quite substantially between different JTOT and IBLOCK values, so that an energy range that is appropriate for one case may not be appropriate for another. In such cases BOUND and FIELD are usually used for one JTOT and IBLOCK at a time, with JTOTU = JTOTL (and often IBFIX > 0, IBHI = 0). These choices are also often suitable in MOLSCAT when searching for and characterising energy-dependent resonances in the eigenphase sum (section 9.9) or field-dependent resonances in the scattering length (section 9.10.3) and convergence testing (section 9.5).

4.10 Contracting the basis set

If RCTRCT > 0, BOUND calculates and diagonalises the Hamiltonian matrix at the fixed distance RCTRCT, and then discard eigenvectors of this matrix whose eigenvalues are greater than ECTRCT. This contracted basis set is used in the remainder of the calculation.

This has not proved a particularly useful approach.

4.11 IOS calculations

IOS calculations do not use basis sets constructed from monomer rotational functions. Instead calculations are done for fixed intermolecular angles and the resulting fixed-orientation terms [47, 25] are integrated with appropriate angular factors by quadrature to obtain the collision dynamics factors and hence the state-to-state cross sections. If set greater than 0, LMAX and MMAX specify the highest L and M values for which generalised IOS cross sections, $Q(L, M, M')$ are accumulated. For ITYPE = 103 (section 4.5.3), LMAX and MMAX identify the maximum L for rotors 1 and 2 respectively.

IOSNGP is an integer array of dimension 3. It specifies the number of (Gauss) integration points to use for quadrature over fixed-orientation cross sections in an IOS calculation. IOSNGP(1) is the number of points for θ. In ITYPE=105 and 106, IOSNGP(2) is the number of points for χ; in ITYPE=103, the 3 values are for θ_1, θ_2 and $\phi_1 - \phi_2$.

If values are not given, the program tries to choose the minimum number needed for requested values of LMAX, MMAX and/or input basis set rotor levels.

IPHIFL controls the type of numerical quadrature on ϕ for ITYPE = 103, 105 and 106. The default of 0 requests equally spaced points (recommended; algebraically equivalent to Gauss-Chebyshev quadrature on $\cos\phi$) while IPHIFL ≠ 0 requests Gauss-Legendre.

4.12 The BCT Hamiltonian

The Bohn-Cavagnero-Ticknor (BCT) Hamiltonian [48] provides a simple model for the collision of two species that interact through a dipole-dipole potential. It assumes that the dipoles have a fixed direction in space, at an angle θ to the interparticle vector. The resulting interaction is anisotropic, coupling different partial waves L. The resulting coupled equations are closely analogous to those for ITYPE = 21; they differ only in that the centrifugal Hamiltonian is different and that the "linear molecule" rotational quantum number j in MOLSCAT and BOUND is interpreted as the BCT quantum number L. Use of this Hamiltonian is invoked by setting the logical variable BCT to .TRUE. and BE to a tiny value (such as 1.D-99). The interaction potential for the BCT Hamiltonian contains terms for $\lambda = 0$ and 2 only.

Examples of bound-state calculations with BOUND, using Hamiltonians of this form, are given by Karman *et al.* [49].

4.13 Plug-in basis-set suites (ITYPE = 9)

The programs include an interface for users to specify interaction types different from the built-in ones. A considerable number of such routines have been written, though the interface has developed and become more sophisticated over the years and some older routines would need work to update and test.

In this distribution we provide plug-in basis-suites for two interaction types of current interest:

`base9-1S_3Sigma_cpld.f` handles interactions of a structureless (^1S) atom and a molecule in a $^3\Sigma$ state in a magnetic field;

`base9-alk_alk_ucpld.f` handles interactions of two alkali-metal atoms in ^2S states in a magnetic field, including hyperfine coupling.

These basis-set suites are described in chapter 18; the description of each of them is divided approximately into sections for *users* of the suites and additional sections for people who wish to understand their internal working, perhaps in order to program their own basis-set suite for a different case.

Chapter 17 gives a more formal description of the components of a plug-in basis-set suite, and specifies the calling sequence of each routine involved. It is intended primarily for people who wish to program their own basis-set suite.

Plug-in basis set suites usually have access to many variables input in namelist &BASIS, which are placed in module basis_data and are listed in section 17.12.1. These can be used as desired by the programmer.

Chapter 5

Constructing the interaction potential

5.1 The potential expansion

The interaction potential $V(R, \xi_{\text{intl}})$ is formally a function of all the coordinates. However, it has the additional property that it is invariant to rotations of the whole system. In practice interaction potentials are usually written as a function of relative coordinates, with angles expressed with respect to the interparticle vector $\boldsymbol{R}$.

The programs internally require an expansion of the interaction potential in a set of orthogonal functions of the internal coordinates*,

$$V(R, \xi_{\text{intl}}) = \sum_{\Lambda} v_{\Lambda}(R) \mathcal{V}^{\Lambda}(\xi_{\text{intl}}), \tag{5.1}$$

where the labels that comprise Λ depend on `ITYP` and are held in an array `LAMBDA`.

Interaction type		Λ	notes
linear rigid rotor + atom	1	λ	
linear vibrotor + atom	2	$\lambda\, v\, v'$	$\lambda\, v\, v' \equiv \lambda\, v'\, v$; only one should be supplied
linear rigid rotor + linear rigid rotor	3	$\lambda_1\, \lambda_2\, \lambda$	$\lambda_1\, \lambda_2\, \lambda \equiv \lambda_2\, \lambda_1\, \lambda$; both *must* be supplied
non-linear rigid rotor + linear rigid rotor	4	$\lambda_1\, \kappa_1\, \lambda_2\, \lambda$	
non-linear rigid rotor + atom	5 & 6	$\lambda\, \kappa$	
linear vibrotor + atom	7	$\lambda\, v\, j\, v'\, j'$	$\lambda\, v\, j\, v'\, j' \equiv \lambda\, v'\, j'\, v\, j$; only one should be supplied
atom + surface	8	$g_1\, g_2$	

The terms included in the potential expansion, and values for the radial potential coefficients

*This is not true for the IOS code in MOLSCAT, which can use unexpanded potentials directly; see section 5.4.

$v_\Lambda(R)$, are provided by subroutine POTENL. Versions of the programs before 2019.0 were documented in the expectation that POTENL would usually be a plug-in routine. This is still possible, as described in chapter 16 below. However, the programs are now supplied with a general-purpose version of POTENL that is adequate for most purposes. This section documents the capabilities of the general-purpose version of POTENL.

The radial potential coefficients may be generated in two different ways.

1. Coefficients supplied explicitly. Very simple coefficients (sums of exponential terms and inverse-power terms) can be set up in POTENL itself. More complicated coefficients must be provided by a user-supplied routine VINIT (with entry points VSTAR, VSTAR1 and VSTAR2) as described in section 5.3.1 below.

2. For most built-in interaction types, a user-supplied routine VRTP may be provided to evaluate the interaction potential at values of the coordinates specified by POTENL, as described in section 5.3.2 below. POTENL then evaluates the radial potential coefficients itself using appropriate quadratures.

Both VINIT/VSTAR and VRTP have much simpler specifications than POTENL itself.

On initialisation, POTENL reads namelist &POTL and returns integer labels for the potential expansion terms in the array LAMBDA. These labels are usually generated from namelist &POTL items LMAX, MMAX, L1MAX and L2MAX (as appropriate for the value of ITYP), but in special cases the LAMBDA array may be specified as an explicit list. In this case MXLAM must be the number of expansion functions.

The type of functions used for expanding the interaction potential depends on ITYP:

5.1.1 Rigid rotor + atom (ITYP = 1)

The interaction potential is expanded in Legendre polynomials

$$V(R,\theta) = \sum_\lambda v_\lambda(R) P_\lambda(\cos\theta). \tag{5.2}$$

Note that the Legendre polynomials P_λ are normalised with $P_\lambda(1) = 1$, so that they are orthogonal but not orthonormal.

If MXLAM = 0 (the default), LMAX sets the maximum value for λ, and POTENL creates a LAMBDA array containing entries for all non-zero terms from $\lambda = 0$ upwards. Setting IHOMO = 2 creates a LAMBDA array that contains only even values of λ; this is appropriate if the molecule has symmetry $D_{\infty h}$ (e.g., if it is a homonuclear diatomic molecule).

If MXLAM > 0, the array LAMBDA is read as an explicit list of MXLAM values.

5.1.2 Diatomic vibrotor + atom (ITYP = 2 or 7)

The potential matrix element between each pair of molecular vibration-rotation functions is expanded in Legendre polynomials

$$\langle v'j'|V(R,r,\theta)|vj\rangle = \sum_{\lambda} v_{\lambda vjv'j'}(R)P_{\lambda}(\cos\theta). \tag{5.3}$$

The Legendre polynomials P_λ are normalised as for ITYP = 1. For ITYP = 7 each term in the expansion is represented by 5 consecutive elements of the LAMBDA array: λ, v, j, v', j'. For ITYP = 2 the effect of monomer centrifugal distortion on the potential matrix elements is neglected; the j labels are omitted, and each term is represented by 3 indices: λ, v, v'.

If ITYP = 2 and MXLAM = 0 (the default), LMAX sets the maximum value for λ, and IHOMO = 2 may be used, as for ITYP = 1. The vibrational quantum numbers v and v' loop from IVMIN to IVMAX.

For ITYP = 7, MXLAM must be set greater than zero and the array LAMBDA must be provided as an explicit list.

If MXLAM > 0, the array LAMBDA is read as an explicit list of MXLAM triples λ, v, v' (for ITYP = 2) or quintuples λ, v, j, v', j' (for ITYP = 7). The interaction potential is invariant to exchange of the labels v and v' (or v, j and v', j'), and only one should be supplied.

5.1.3 Rigid rotor + rigid rotor (ITYP = 3)

The interaction potential is expanded in coupled products of spherical harmonics for the two molecules [20],

$$V(R,\theta_1,\phi_1,\theta_2,\phi_2) = \sum_{\lambda_1,\lambda_2,\lambda} v_{\lambda_1,\lambda_2,\lambda}(R) \sum_{\mu} \langle \lambda_1\mu, \lambda_2, -\mu|\lambda, 0\rangle \left(\frac{2\lambda+1}{4\pi}\right)^{1/2} \times Y^{\mu}_{\lambda_1}(\theta_1,\phi_1)Y^{-\mu}_{\lambda_2}(\theta_2,\phi_2). \tag{5.4}$$

The interaction potential actually depends only on the relative angle $\phi_1 - \phi_2$. The spherical harmonics Y^{μ}_{λ} are normalised by integrating over angles, so the set of functions is orthonormal.[†]

The coupled expansion is not the only one commonly in the literature. An alternative is an uncoupled expansion,

$$V(R,\theta_1,\phi_1,\theta_2,\phi_2) = \sum_{\lambda_1,\lambda_2,\mu} v'_{\lambda_1,\lambda_2,\mu}(R) \left(\frac{[(2\lambda_1+1)(2\lambda_2+1)]^{1/2}}{4\pi}\right) Y^{\mu}_{\lambda_1}(\theta_1,\phi_1)Y^{-\mu}_{\lambda_2}(\theta_2,\phi_2). \tag{5.5}$$

The uncoupled expansion is not directly implemented in these programs, but coefficients may readily be converted between the two forms. The coupled expansion has the advantage that

[†]This normalisation differs from that for the Legendre polynomials used for ITYP = 1, 2 and 7, and has the important consequence that $v_{000}(R)$ is *not* the spherical average of the potential.

electrostatic interactions between multipole moments on the two monomers each result in a single long-range term.

Each term in the expansion is represented by 3 consecutive elements of the LAMBDA array: λ_1, λ_2 and their vector sum λ.

If MXLAM $= 0$ (the default), L1MAX and L2MAX set the maximum values for λ_1 and λ_2, and λ takes all integer values from $|\lambda_1 - \lambda_2|$ to $\lambda_1 + \lambda_2$. IHOMO and/or IHOMO2 can be set to 2 to indicate that the corresponding molecule has symmetry $D_{\infty h}$.[‡]

If MXLAM > 0, the array LAMBDA is read as an explicit list of MXLAM triples $\lambda_1, \lambda_2, \lambda$.

If the two rotors are identical, potential terms $v_{\lambda_1,\lambda_2,\lambda}(R)$ and $v_{\lambda_2,\lambda_1,\lambda}(R)$ must be identical and both must be included.

5.1.4 Non-linear molecule + atom (ITYP $= 5$ or 6)

The interaction potential is expanded in spherical harmonics [24],[§]

$$V(R,\theta,\chi) = \sum_{\lambda,\kappa} v_{\lambda,\kappa}(R) Y_\lambda^\kappa(\theta,\chi). \tag{5.6}$$

ITYP $= 5$ or 6 are coded only for molecules for which the molecule-fixed xz plane is a plane of symmetry, and in this case the expansion becomes

$$V(R,\theta,\chi) = \sum_{\lambda,\kappa\geq 0} v_{\lambda,\kappa}(R)(1+\delta_{\kappa,0})^{-1}\left[Y_\lambda^\kappa(\theta,\chi) + Y_\lambda^{-\kappa}(\theta,\chi)\right]. \tag{5.7}$$

Each term in the expansion is represented by 2 consecutive elements of the LAMBDA array: λ and $|\kappa|$.

If MXLAM $= 0$ (the default), LMAX and MMAX set the maximum values for λ and $|\kappa|$. If ICNSYM > 0, only terms where $|\kappa|$ is a multiple of ICNSYM are included; this is appropriate for molecules where the z axis is a proper axis of rotation of order ICNSYM.

If MXLAM > 0, the array LAMBDA is read as an explicit list of MXLAM pairs λ, $|\kappa|$.

The interaction potential may also be expanded in terms of renormalised (Racah-normalised) spherical harmonics $C_{\lambda\kappa}(\theta,\chi) = [4\pi/(2\lambda+1)]^{1/2} Y_\lambda^\kappa(\theta,\chi)$ in place of the normalised spherical harmonics $Y_\lambda^\kappa(\theta,\chi)$. This form of expansion is indicated by setting CFLAG to 1.

5.1.5 Asymmetric rotor + diatom (ITYP $= 4$)

The interaction potential is expanded in coupled products of rotation matrices for the asymmetric rotor and spherical harmonics for the diatom [23]. This is a generalisation of the expansion for ITYP $= 3$, with an extra angle χ representing the rotation of the asymmetric

[‡] in this particular instance ICNSYM is a deprecated synonym for IHOMO2, retained for backwards compatibility.

[§] This normalisation differs from that for the Legendre polynomials used for ITYP $= 1$, and has the important consequence that $v_{00}(R)$ is *not* the spherical average of the potential.

rotor about its z axis. The spherical harmonics for the asymmetric rotor are replaced by normalised rotation matrices, with an index κ_1 corresponding to the angle χ.

Each term in the expansion is represented by 4 consecutive elements of the LAMBDA array: λ_1 (the tensor order of the expansion for the non-linear molecule), κ_1 (the projection of λ_1 on the z axis of the non-linear molecule), λ_2 (the tensor order of the expansion for the diatom) and λ (the vector sum of λ_1 and λ_2).

There is no option to generate the LAMBDA array from limits supplied in other namelist items. MXLAM must be set greater than 0, and the array LAMBDA is read as an explicit list of MXLAM sets of 4 consecutive values λ_1, κ_1, λ_2 and λ.

5.1.6 Atom + corrugated solid surface (ITYP = 8)

The interaction potential is expanded in surface reciprocal lattice vectors,

$$V(\boldsymbol{r}) = V(\boldsymbol{R}, z) = \sum_{g_1,g_2} v_{\boldsymbol{G}}(z) \exp(\mathrm{i}\boldsymbol{G} \cdot \boldsymbol{R}). \tag{5.8}$$

LAMBDA has 2 entries per term in the expansion: g_1 and g_2, which are the two components of the reciprocal lattice surface vector $\boldsymbol{G}$. The general-purpose version of POTENL does not generate the LAMBDA array from limits supplied in other namelist items, and it must be input as an explicit list of MXLAM pairs (g_1, g_2).

The interaction potential may sometimes be described in terms of a sum of pairwise interactions between the colliding atom and the particles that make up the surface:

$$V(\boldsymbol{r}) = \sum_j U(|\boldsymbol{r} - \boldsymbol{r}_j|), \tag{5.9}$$

where $\boldsymbol{r}_j$ is the jth lattice site. This gives for the Fourier components of the atom-surface potential

$$v_{\boldsymbol{G}}(z) = \frac{1}{a_c} \int \sum_j U(|\boldsymbol{r} - \boldsymbol{r}_j|) \exp(-\mathrm{i}\boldsymbol{G} \cdot \boldsymbol{R})\, d^2\boldsymbol{R}, \tag{5.10}$$

where a_c is the area of the direct-space unit cell.

5.2 Units of length and energy for the interaction potential

The quantities RM and EPSIL specify the units of length and energy used by the potential routine. They are specified in namelist &POTL in units of Ångstrom and cm^{-1} respectively.

Any values of RM and EPSIL set in namelist &POTL are passed to the user-supplied routine VINIT (if LVRTP is set .FALSE.) or VRTP (if LVRTP is set .TRUE.) and are overwritten if those routines change their values.

If RM is not specified, it defaults to the value of RUNIT in namelist &INPUT. If RUNIT is not specified, RM is used as the unit of length throughout the programs. If neither is specified, both are set to 1.0 (indicating length units of Å).

5.3 Evaluating the radial potential coefficients

5.3.1 Potential pre-expanded in internal coordinates

If the radial potential coefficients are to be supplied explicitly (as opposed to generated by quadrature in POTENL), the array NTERM sets out how they are to be calculated:

- If NTERM(i) ≥ 0, the ith coefficient is evaluated as a sum of NTERM(i) separate terms, each of which is either an exponential in R or an inverse power of R. The function for each of these terms is specified by the arrays NPOWER, E and A, as described below.
- If NTERM(i) < 0, POTENL obtains the coefficient by a call to VSTAR, as described below.

Expansion coefficients supplied as data in &POTL

The arrays NPOWER, E and A are interpreted as in the code fragment:[¶]

```
IEXP  = 0
ITERM = 0
DO I = 1, MXLAM
  P(I) = 0.D0
  IF (NTERM(I).LT.0) CALL VSTAR(R, I, P(I))
  DO JTERM = 1, NTERM(I)
    ITERM = ITERM + 1
    IF (NPOWER(ITERM).NE.0) THEN
      P(I) = P(I) + A(ITERM) * R**NPOWER(ITERM)
    ELSE
      IEXP = IEXP + 1
      P(I) = P(I) + E(IEXP) * EXP(R * A(ITERM))
    ENDIF
  ENDDO
ENDDO
```

For example, consider a potential with 3 expansion terms:

- the first is provided by a user-supplied VINIT/VSTAR subroutine;
- the second is a Lennard-Jones potential $C_{12}R^{-12} - C_6R^{-6}$;
- the third is an exponential-6 potential $a\exp(-\beta R) - bR^{-6}$.

In this case, the input file would contain: NTERM $= -1,\ 2,\ 2$, NPOWER $= -12,\ -6,\ 0,\ -6$, A $= C_{12}, -C_6, a, -b$ and E $= -\beta$.

Expansion coefficients supplied by VINIT/VSTAR

If any of the elements of NTERM are negative, user-supplied subroutines VINIT, VSTAR, VSTAR1 and VSTAR2 must be provided. It is simplest to provide the latter 3 as entry points to VINIT.

[¶] In versions before 2019.0, any positive values of NPOWER were converted to negative values. Positive values are now allowed and left unchanged.

VINIT is called once for each term in the potential expansion for which NTERM(i) < 0. Its purpose is to read any data required and to carry out any R-independent processing desired, to save time on future calls. Its specification is

```
SUBROUTINE VINIT(I, RM, EPSIL)

DOUBLE PRECISION, INTENT(INOUT) :: RM, EPSIL
INTEGER,          INTENT(IN)    :: I
```

The index of the potential expansion term is passed in I. The current values of the length units factor RM and the energy units factor EPSIL are passed in as arguments, and may be overwritten if desired. Calls to VINIT for different values of I should not return different values of RM or EPSIL.

If desired, VINIT may supply names for the units RM and EPSIL in the character variables RMNAME and EPNAME in module potential. These variables are pre-populated with 'RM' and 'EPSIL' respectively, but if RUNIT $= 1.0$ after the call to POTENL, the programs set RUNAME to 'ANGSTROM'; similarly, if EPSIL $= 1.0$, they set EPNAME to 'CM-1'.

The entry points (or subroutines) VSTAR, VSTAR1 and VSTAR2 are subsequently called many times to evaluate the radial potential coefficients for either the interaction potential or its first or second radial derivatives respectively. The specification of these routines is

```
SUBROUTINE VSTAR(I, R, V)

DOUBLE PRECISION, INTENT(IN)  :: R
DOUBLE PRECISION, INTENT(OUT) :: V
INTEGER,          INTENT(IN)  :: I
```

and similarly for VSTAR1 and VSTAR2 (but see the next paragraph). The argument R contains the distance R (in units of RM) and the routine must return the value of the radial potential coefficient I (or its first or second radial derivative) in the argument V, in units of EPSIL.

If required, VINIT and VSTAR can access the LAMBDA array by using the module potential, described in section 17.12.2. This facility is particularly useful if the ordering of the potential terms (or which ones are needed) depends on the basis set.

The only functions of the current programs that use radial potential derivatives are the VIVS propagator and calculations of nonadiabatic matrix elements at high print level in the LDMA propagator. Even for these, derivatives may be calculated numerically by specifying NUMDER=.TRUE. in namelist &INPUT. For all other purposes it is adequate to supply VSTAR1 and VSTAR2 routines that prints an error message and stop if they are called.

5.3.2 Potential supplied by VRTP as function of internal coordinates

It is often more convenient to specify the interaction potential $V(R, \xi_{\mathrm{intl}})$ explicitly as a function of internal coordinates ξ_{intl}, rather than as an expansion over internal functions. In

such cases LVRTP must be set to .TRUE. and the user-supplied routine VRTP must supply values for the interaction potential at a given set of internal coordinates.

The expansion of the interaction potential in terms of sets of orthogonal functions can be inverted to give

$$v_\Lambda(R) = N_\lambda \int d\xi_{\text{intl}}\, V(R, \xi_{\text{intl}}) \boldsymbol{\mathcal{V}}^\Lambda(\xi_{\text{intl}}), \tag{5.11}$$

where N_λ depends on the normalisation of the expansion functions (and is 1 if they are orthonormal). The integral can be approximated by numerical quadrature

$$v_\Lambda(R) = N_\lambda \sum_{i=1}^{n} w_i \boldsymbol{\mathcal{V}}^\Lambda(\xi_i) V(R, \xi_i), \tag{5.12}$$

where ξ_i and w_i are the n points and weights required for the quadrature scheme over the functions $\boldsymbol{\mathcal{V}}^\Lambda(\xi)$.

The quadrature scheme is a combination of one or more of the following types of quadrature over each individual internal coordinate.

type of quadrature	x_i	w_i
Gauss-Legendre	ith root of $P_n(x)$	$2(P_n(1))^2/[(1-x_i^2)(P_n'(x_i))^2]$
Gauss-Hermite	ith root of $H_n(x)$	$2^{n-1}n!\sqrt{\pi}/[n^2(H_{n-1}(x_i))^2]$
Equally spaced	$[2i-1]\pi/2n$	π/n

POTENL calculates the points and weights required for each type of Gaussian quadrature needed for the interaction types 1, 2, 3, 5 and 6.

ITYP	NPTS(1) (or NPT)	NPTS(2) (or NPS)	NPTS(3)
	quadrature scheme	quadrature scheme	quadrature scheme
1	Gauss-Legendre over $\cos\theta$		
2	Gauss-Legendre over $\cos\theta$	Gauss-Hermite over r	
3	Gauss-Legendre over $\cos\theta_1$	Gauss-Legendre over $\cos\theta_2$	equally spaced over $\phi_1 - \phi_2$
5 & 6	Gauss-Legendre over $\cos\theta$	equally spaced over χ	

Note that Gauss-Legendre quadrature is not optimum for the integral over $\cos\theta$ for symmetric top functions with $k > 0$, but it is still used.

The number of points required for these quadrature schemes may be given in NPTS. It should be at least $\lambda + 1$ (Gauss-Legendre quadrature), $\mu + 1$ or $\kappa + 1$ (equally spaced) or $v + v' + 1$ (Gauss-Hermite). If the value supplied for each quadrature is lower than this, POTENL prints a warning and increases it to the minimum valid value.

If IHOMO $= 2$, only points for $\theta \geq 90°$ are used. If ICNSYM > 1, NPTS(2) points are used in the range $0 < \phi_1 - \phi_2 < \pi/$ICNSYM.

Subroutine VRTP must be supplied by the user. Its specification is:

```
SUBROUTINE VRTP(IDERIV, R, V)

USE angles

DOUBLE PRECISION, INTENT(INOUT) :: R
DOUBLE PRECISION, INTENT(OUT)   :: V
INTEGER,          INTENT(IN)    :: IDERIV
```

On an initialisation call, IDERIV $= -1$. The routine may read any data required and carry out any coordinate-independent processing desired, to save time on future calls. It may also specify the units of length and energy to be used in subsequent calls by overwriting the variables RM (= R) and EPSIL (= V). It may also set the variables IHOMO, IHOMO2, ICNSYM, and ICNSY2 in module angles; see below for specification.‖

If desired, an initialisation call to VRTP may supply names for the units RM and EPSIL in the character variables RMNAME and EPNAME in module potential.

Subsequent calls specify IDERIV $= 0$, 1, 2 for the interaction potential and its first and second derivatives respectively. The distance (in units of RM) is specified in argument R whilst the internal coordinates are supplied in the COSANG array in module angles (see below). VRTP must return the corresponding interaction potential (in units of EPSIL) in argument V.

The only functions of the current programs that use radial potential derivatives are the VIVS propagator and calculations of nonadiabatic matrix elements at high print level in the LDMA propagator. Even for these, derivatives may be calculated numerically by specifying NUMDER=.TRUE. in namelist &INPUT. For all other purposes it is adequate to supply a VRTP routine that traps calls with IDERIV > 0, prints an error message, and stops.

For ITYP $= 1$, 2, 3, 5, and 6, COSANG(1) is $\cos\theta$.

For ITYP $= 3$, COSANG(2) is $\cos\theta_2$ and COSANG(3) is $\phi_1 - \phi_2$.

For ITYP $= 5$ and 6, COSANG(2) is χ.

For non-IOS ITYP $= 2$ cases, COSANG(2) is q, the reduced harmonic oscillator coordinate, such that the ground-state vibrational wavefunction of the diatom is $\exp(-q^2/2)$; the quadrature code in POTENL uses *harmonic* vibrational wavefunctions for the vibrating diatomic molecule. If a more sophisticated integration is required, it must be programmed separately (e.g., in VINIT).

Module angles

The specification of module angles is

```
USE sizes, ONLY: MXANG
INTEGER          IHOMO, ICNSYM, IHOMO2, ICNSY2
DOUBLE PRECISION COSANG(MXANG), FACTOR
```

‖Before version 2019.0, the quantities included in module angles were in a common block named ANGLES. Earlier versions of VRTP must be modified to use module angles.

The array dimension MXANG is set in module sizes and is currently 7.

5.4 IOS calculations

IOS calculations perform propagations at fixed orientations. This is most naturally done by setting LVRTP = .TRUE. to use the VRTP method described above. There is no need to obtain radial potential coefficients by numerical quadrature, so MXLAM is reset internally to 1, and the quantum labels in LAMBDA are ignored.**

For ITYPE = 102, VRTP does not use COSANG(2), and must return in argument V a vector of matrix elements $\langle v'|V(R,\theta)|v\rangle$ in the order expected by POTENL. MXLAM must be set to a negative value, with v and v' values supplied as LAMBDA($3i-1$) and LAMBDA($3i$) for $i = 1, |\text{MXLAM}|$. LAMBDA($3i-2$), which would normally be the order of the Legendre polynomial, is not used.

IHOMO and ICNSYM would normally be determined automatically by examining the LAMBDA array, but this can be done only if the interaction potential is expanded in angular functions. They may be set explicitly in the input file if desired.

If LVRTP = .FALSE., the terms of the potential expansion should be indicated as for non-IOS calculations and the VINIT/VSTAR mechanism may be used if desired.

**In versions before 2019.0, when LVRTP = .TRUE. and MXLAM > 0, MOLSCAT projected out the potential expansion terms using quadrature, and then re-summed them to obtain the interaction potential at each orientation. This was inefficient and unnecessarily approximate.

Chapter 6

Specifying input energies

For each total angular momentum JTOT and symmetry block IBLOCK, the programs loop over energies and external fields. The coupling matrices for external fields must be implemented in plug-in basis-set suites and are described in chapter 7 below. This chapter describes control of the energy. For MOLSCAT this is the energy for the scattering calculation. For BOUND or FIELD it is the proposed energy for a bound state.

The energy may optionally be specified with respect to the energy of separated monomers in specified states (a scattering threshold). We therefore include here a description of how to interpret the indices used to identify scattering channels in the output, to assist both in choosing the right index and in identifying S-matrix elements and quantities obtained from them.

6.1 Units of energy

The units of energy for quantities input in &INPUT and for most output energies are *independent* of those used for quantities in &BASIS. They are specified by EUNITS, which is an integer that selects a unit of energy from the list in section 3.7. The default for EUNITS is 1, indicating energies expressed as wavenumbers in cm^{-1}.

If the energy unit required is not among those listed, EUNITS may be set to 0 and the required value (in units of cm^{-1}) supplied in EUNIT. If this is done, the name of the unit should be supplied in the character variable EUNAME.

EUNITS, EUNIT and EUNAME are distinct from the variables with the same names in namelist &BASIS (section 4.4), but their allowed values and interpretation are the same.

The input quantities affected by EUNITS or EUNIT are ENERGY, DNRG, EMIN and EMAX, EREF, the (energy) convergence criteria DTOL (for BOUND only), and DEGTOL.

The input energies are converted immediately into cm^{-1} using the appropriate conversion factor. The programs often give energies in both cm^{-1} and in the units specified by EUNITS. Any energies in the output that are not given an explicit unit are in cm^{-1}.

6.2 Specifying energies for MOLSCAT and FIELD calculations

The total energies at which calculations are performed are controlled by the array ENERGY and the variables NNRG, DNRG and LOGNRG. Calculations are performed for NNRG energies. If DNRG $\neq 0.0$, calculations are performed at NNRG equally spaced energies, DNRG apart, starting at ENERGY(1). If DNRG $= 0.0$ and LOGNRG = .FALSE., the NNRG energies must be supplied in the ENERGY array. If DNRG $= 0.0$ and LOGNRG = .TRUE. the NNRG energies are geometrically spaced between ENERGY(1) and ENERGY(2). The maximum allowed value of NNRG is set in the variable MXNRG, which is 2000 in MOLSCAT and 100 in FIELD.

6.3 Internally generated energies in MOLSCAT

There are special uses of the NNRG and ENERGY that cause MOLSCAT to generate energy lists internally for:

1. Searching for energy-dependent resonances in the S-matrix eigenphase sum;
2. Calculating line-broadening cross sections.

These are described separately in sections 9.9 and 9.7 respectively.

6.4 Temperatures for thermal averaging

An alternative form of input is available to facilitate thermal averaging of cross sections using Gaussian quadrature. This is controlled by the array TEMP and the variables NTEMP and NGAUSS. If NTEMP > 0, MOLSCAT calculates appropriate energies and weighting factors which correspond to NGAUSS-point Gaussian quadratures for each of the NTEMP different temperatures (in Kelvin) in the TEMP array. Scattering calculations are then performed at each of the NGAUSS $\times$ NTEMP energies. The maximum allowed values are NTEMP $= 5$ and NGAUSS $= 6$.

The thermal averaging itself must be done outside MOLSCAT. Note that Gaussian quadrature is not a reliable way of thermally averaging some types of cross sections, particularly if there are resonances present, and use of this option is generally not recommended for precise work.

6.5 Specifying energies for BOUND calculations

The energies at which calculations are performed in BOUND are governed by the namelist items EMAX and EMIN. EMAX must be greater than EMIN. BOUND calculates the node counts at EMAX and EMIN. The difference between them is the number of states in the energy interval. By default, BOUND attempts to locate all these states. However, if NODMAX and/or NODMIN are also set non-zero, only states between EMIN and EMAX with node counts in the range NODMIN to NODMAX are located.

6.6 The reference energy

By default, the zero of energy used for total energies is the one used for monomer energies, as described in chapter 4, or defined by the monomer Hamiltonians programmed in a plug-in basis-set suite. However, it is sometimes desirable to use a different zero of energy (reference energy), such as the energy of a particular scattering threshold (which may depend on external fields). This energy can be specified in several different ways:

- If EREF $\neq 0$, it is used as the reference energy for all scattering and bound-state energies.
- If $H_{\rm intl}$ is diagonal (which includes all the built-in coupling cases and plug-in basis-set suites with NCONST $= 0$):
  - If MONQN$(1) = -99999$ (the default) and IREF > 0, the programs use the energy of the pair level with index IREF.
  - If MONQN$(1) \neq -99999$, the values supplied in the array MONQN are quantum labels for the reference threshold, which must match those in a row of the array JLEVEL.
- If $H_{\rm intl}$ is non-diagonal (which includes plug-in basis-set suites with NCONST > 0):
  - If MONQN$(1) = -99999$ (the default) and IREF > 0, the programs use the threshold energy of the channel with index IREF. The user must identify the index of the required channel before the full calculation (which often requires a pilot calculation; see section 6.6.1 below).
  - If MONQN$(1) \neq -99999$, the code requires a routine THRSH9 as part of a plug-in basis-set suite to calculate the reference energy from the values in the array MONQN. See section 17.10.2 for further details.

6.6.1 Specifying the index of the reference threshold

For the built-in interaction types (ITYP $\neq 9$), or for plug-in basis-set suites that use a basis set in which $H_{\rm intl}$ is diagonal, it is almost always easiest to specify the reference threshold via the array MONQN.

Some older plug-in basis-set suites for basis sets in which $H_{\rm intl}$ is non-diagonal do not implement the routine THRSH9, so it is not possible to specify a reference energy via the array MONQN. In this case it may be desired to specify the reference energy via a threshold index IREF.

It is usually necessary to carry out a pilot calculation to identify the index required. If $H_{\rm intl}$ is non-diagonal, the programs print a list of threshold energies calculated from the internal Hamiltonian if IPRINT ≥ 10. Any threshold index with the required threshold energy may be specified; the value of L is immaterial.

The IREF mechanism is intended for calculations where there is no loop over JTOT or IBLOCK, and does *not* work in cases where the threshold required does not exist for the first JTOT/IBLOCK combination. In complicated cases it may be quite involved to identify the index required, and under these circumstances it is often better to use (or implement) a THRSH9 routine in the basis-set suite, so that the reference energy can be specified via MONQN instead.

Chapter 7

External fields and scaling the interaction potential

MOLSCAT and BOUND were extended in 2007 to incorporate the effects of external magnetic and/or electric fields [50]. Version 2019.0 introduced a more general structure that allows multiple external fields (which may be static, such as electric or magnetic fields, or oscillatory, such as photon fields). None of the built-in interaction types include any external fields, so they must be implemented within a plug-in basis-set suite for `ITYP` = 9. The current release includes two examples of plug-in basis-set suites that include external magnetic fields. If the user wishes to take advantage of this feature, they should also refer to the specification of the module `efvs` in section 17.12.3 in order to work out exactly what they need to program.

Each calculation is carried out with external fields specified by `NEFV` real numbers `EFV`, referred to here as external field variables (EFVs). These can include field strengths, frequencies, relative angles, or other variables. The programmer of the plug-in basis-set suite specifies `NEFV` and may use the elements of `EFV` in any way desired. The programs normally consider all but one of the EFVs as fixed, but allow one of them (identified by the index `IFVARY`) to be varied. In the simplest case the variation is specified as a grid, but MOLSCAT and FIELD can locate resonances or bound states, respectively, as a function of the varying EFV.

The interaction potentials that are used for coupled-channel calculations are seldom known exactly. Exploring the sensitivity of calculated properties to parameters of the interaction potentials is a complicated task, but some estimate may be obtained by scaling the potential. Ultracold scattering properties often show extreme sensitivity to such variations. The programs allow such a scaling to be incorporated by treating it as an artificial EFV; MOLSCAT can converge on resonances as a function of potential scaling, and FIELD can find bound states as a function of potential scaling. By default the entire potential is scaled by the same factor, but the subroutine that performs the scaling (`SCAPOT`) can be replaced with a bespoke version that can apply a different scaling to each term in the potential expansion.

7.1 Using external fields

The items in namelist &INPUT that are used to control the values and types of external field variables included, are:

IFVARY (deprecated synonym MAGEL) specifies the index of the EFV to be varied in the array EFV. If it is 0, the potential scaling factor is varied.

NFVARY specifies the number of varying EFVs included in the FIELD array.
NFVARY is usually 0 or 1. NFVARY > 1 is implemented *only* for varying EFVs supplied explicitly as a list of values in the array FIELD. In this case IFVARY is a list of indices of the variable EFVs.

FIXFLD is an array of dimension NEFV (limited by MXEFV = 10, set in module efvs). It specifies values of all the EFVs in the array EFV; the value for element IFVARY is ignored, so FIXFLD is not needed if NEFV is 1.

FLDMIN is the lower end of the range of the (single) varying EFV.

FLDMAX is the upper end of the range of the (single) varying EFV.

DFIELD is the step size between values of the (single) varying EFV. In MOLSCAT, DFIELD is ignored if a resonance is to be characterised (IFCONV > 0, section 9.10.3) or a specific value of the scattering length is to be located (section 9.10.4). In FIELD, DFIELD (default 10^{30}) must be greater than FLDMAX − FLDMIN if bound states are to be located.

DTOL is the convergence criterion used when converging on quantities as a function of the (single) varying EFV: on resonances or values of the scattering length/volume in MOLSCAT; or on the position of bound states in FIELD. (In BOUND, DTOL is the convergence criterion for the energy of the bound state).

NFIELD If both FLDMIN and FLDMAX are 0.0 (the default), NFIELD sets of values of EFV must be supplied in the FIELD array (but only for the EFVs whose indices were given in IFVARY).

FIELD is an array of dimension MXFLD (= 10000) containing values of EFV for the varying EFVs.

IFIELD (obsolete, but retained for backwards compatibility) specifies the index of the first EFV in the VCONST array in certain plug-in basis-set suites. This quantity is now unnecessary and any value in &INPUT namelist is ignored.

7.2 Potential scaling

The programs implement a potential scaling factor (SCALAM) that scales the whole interaction potential. The scaling factor is handled in the same way as an EFV.

If the scaling factor is to be held fixed while an EFV is varied, it must be specified in SCALAM.

Alternatively, setting IFVARY = 0 instructs the programs to treat the scaling factor as the varying quantity. The scaling factor is handled in the same way as a varying EFV, so that:

FLDMIN is the lower bound value of the scaling factor.

FLDMAX is the upper bound of the scaling factor.

DFIELD is the step size between values of the scaling factor. If a resonance is to be characterised as a function of SCALAM (section 9.10), DFIELD is not used.

DTOL is the convergence criterion used when converging on quantities as a function of the scaling factor: on resonances or values of the scattering length/volume in MOLSCAT; or on the position of bound states in FIELD.

Note that, if NFVARY > 1, including 0 among the indices of varying EFVs in IFVARY means that the grid of values held in the FIELD array include the scaling factor. If, however, the scaling factor is to be held constant at a value other than 1, it must be set in SCALAM, not in the FIXFLD array.

Chapter 8

Controlling the propagators

8.1 Propagator choice

Versions of the programs before 2019.0 implemented a variety of propagators to solve the coupled equations, and allowed some specific combinations of them as "hybrid" propagators that combine a propagator suitable at short range with a different one suitable at long range. From version 2019.0, this mechanism was generalised to allow *any* sensible combination of a short-range propagator with a long-range propagator.

The methods used to propagate solutions to the coupled-channel equations are controlled by IPROPS to specify the short-range propagator and IPROPL to specify the long-range propagator. If only IPROPS is specified, IPROPL is set the same as IPROPS. If neither is specified, the default combinations are IPROPS = IPROPL = 6 for BOUND and FIELD, and IPROPS = 6 and IPROPL = 9 for MOLSCAT.

The propagator codes are:

- **2** de Vogelaere (DV)
- **3** R-matrix (RMAT)
- **4** Variable-interval variable-step (VIVS)
- **5** Johnson log-derivative (LDJ)
- **6** Manolopoulos diabatic modified log-derivative (LDMD)
- **7** Manolopoulos quasiadiabatic modified log-derivative (LDMA)
- **8** Manolopoulos-Gray symplectic log-derivative (LDMG)
- **9** Alexander-Manolopoulos Airy (AIRY)
- **−1** WKB phase integrals by quadrature

If IPROPL = 0, it is set the same as IPROPS.

For backwards compatibility, INTFLG is a deprecated synonym for IPROPS, except that there are two values of INTFLG that have special meanings:

INTFLG = 4 sets IPROPS = 5 and IPROPL = 4, corresponding to the VIVAS hybrid propagator of Parker *et al.* [51].

INTFLG = 8 sets IPROPS = 6 and IPROPL = 9, corresponding to the hybrid LDMD/AIRY propagator of Alexander and Manolopoulos [42, 43].

Further information on the individual propagators is given in section 8.8 below.

8.2 Units of length

The programs operate in units of lengths specified by RUNIT in namelist &INPUT. If RUNIT is unset, it taken from the value returned by the initialisation call to POTENL). In the general-purpose version of POTENL, the value may be input as RM, though this may be overwritten by code in user-supplied VINIT or VRTP routines. If neither RUNIT nor RM is set, the programs operate in length units of Å.

Input length variables that control the propagation (RMIN, RMAX, RMID, RMATCH, DRS, DRL) are in units of RUNIT. RUNIT itself is specified in units of Å. For example RUNIT = 1.0 (the default) indicates that all distances are in Å, while RUNIT = 0.529177210903 (2018 value) indicates that they are in units of the bohr radius (atomic units). Most output quantities with dimensions of length (including scattering lengths, but not cross sections) are output in units of RUNIT.

8.3 Units of reduced mass

The units of the reduced mass URED are specified by MUNIT. MUNIT itself is specified in units of unified atomic mass units m_u (Daltons). For example MUNIT = 1.0 (the default) indicates that URED is in Daltons, while MUNIT = 5.48579909065D-4 (2018 value) indicates that URED is in units of the electron mass m_e (atomic units).

8.4 Internal units of energy

Quantities with dimensions of energy (total energy E, interaction matrix $\mathbf{W}(R)$, etc.) are processed internally as reduced energies $2\mu E/\hbar^2$, with dimensions of [length]$^{-2}$. This reduces the coupled equations to the form (2.4).

Energies expressed as wavenumbers in cm^{-1} may be converted into reduced energies in Å^{-2} by multiplying by MUNIT$\times$URED/BFCT, where BFCT = $[\hbar/(\mathrm{J\ s})]$ / $(4\pi[c/(\mathrm{m/s})]\ [m_u/\mathrm{kg}])$ [m/cm] $[\mathrm{m/Å}]^2$ = 16.85762919164 (2018 value).

The programs use two conversion factors: dividing by CM2RU = (RUNIT)$^2\times$MUNIT$\times$URED/BFCT converts reduced energies in units of (RUNIT)$^{-2}$ into wavenumbers in cm^{-1}, while dividing by EP2RU = CM2RU$\times$EPSIL converts reduced energies into the units EPSIL used for the interaction potential.

8.5 Ranges of propagation

MOLSCAT propagates the coupled equations outwards from R_{min} to R_{max}, with the option to switch propagation method at R_{mid}.

BOUND and FIELD propagate the coupled equations outwards from R_{min} to R_{match} and inwards from R_{max} to R_{match}. The propagation method may be switched at R_{mid}, which *may* be the same as R_{match} but can be different if desired.

R_{min}, R_{max} and R_{mid} are *based on* the input variables RMIN, RMAX, and RMID, but are not necessarily *equal to* them.

8.5.1 Inner limit R_{min}

If the origin is energetically accessible at the energy of the calculation, R_{min} should be zero. If there is an infinite hard wall at short range, it should be placed at the hard wall. Otherwise, it should be far enough into the classically forbidden region at short range that that the wavefunction at $R < R_{\mathrm{min}}$ does not contribute significantly to the calculated quantities. If a WKB boundary condition is used at short range, as described in section 8.10, it is sufficient to place R_{min} slightly further out, but still well inside the inner turning point.

If IRMSET $= 0$ (default 9), R_{min} is set to RMIN.

If IRMSET > 0, the programs obtain R_{min} from a semiclassical estimate of a distance such that the wavefunction amplitude in all channels is less than $10^{-\mathtt{IRMSET}}$ at R_{min}. This estimate is calculated separately for each JTOT, IBLOCK at the highest energy value in ENERGY (for MOLSCAT and FIELD) or at EMAX (for BOUND).

There are some cases where R is not a radial coordinate, such as scattering from a solid surface (ITYPE $= 8$). To accommodate these, R_{min} is allowed to be negative. However, there are certain program features that are clearly inappropriate and should not be used when R can pass through zero, such as step sizes proportional to a non-zero power of R (section 8.6.2).

8.5.2 Outer limit R_{max}

BOUND and FIELD are designed to operate with a classically forbidden region at long range. Under these circumstances, R_{max} should be far enough into the classically forbidden region that the wavefunction at $R > R_{\mathrm{max}}$ does not contribute significantly to the calculated quantities. If a WKB boundary condition is used at long range, as described in section 8.10, it is sufficient to place R_{max} further in, but still well outside the outer turning point.* This may require very large values of R_{max} for near-threshold states.

BOUND and FIELD always set R_{max} to RMAX.

For MOLSCAT, R_{max} should be large enough that both

1. the interaction potential makes no significant contribution to the wavefunction in the open channels outside R_{max}, so that the open-channel wavefunctions are well represented by the Ricatti-Bessel boundary conditions (2.10);

*For calculations with BOUND and FIELD that are designed to locate quasibound states above the lowest threshold, R_{max} should be placed well into the classically forbidden for the channels that support the quasibound states of interest. Under these circumstances there are additional artificial levels that arise from quantisation of the open-channel continua by the boundary conditions at R_{max}; the energies of these artificial levels depend on R_{max}, and they perturb the quasibound states unphysically when they come close to them.

2. R_{max} is well outside the outer turning point in any closed channels that support scattering resonances of interest, so that the wavefunction in those channels is well represented by the closed-channel boundary condition (which may be a WKB boundary condition as described in section 8.10).

Ultracold scattering calculations may require values of R_{max} comparable to the scattering length, which may be very large, particularly near a Feshbach resonance.

An additional consideration is that open-channel matching does not work well if R_{max} is so far inside a centrifugal barrier in an open channel that the wavefunction has decayed far below its asymptotic amplitude. This is not usually an issue in ultracold scattering, but can be important in cross-section calculations that require high L for convergence.

If IRXSET = 0 (the default), MOLSCAT also sets R_{max} to RMAX.

If IRXSET = 1, MOLSCAT calculates the turning point, at every energy E_j in the ENERGY list, for the pure centrifugal potential of every open channel i with $L_i > 0$. This is

$$R_{\mathrm{cent}}^{ij} = \left[\frac{\hbar^2 L_i(L_i+1)}{2\mu(E_j - E_{\mathrm{intl},i})}\right]^{\frac{1}{2}} . \tag{8.1}$$

The value used for R_{max} is the largest of all these values and RMAX.

This approach generally works well for cross-section calculations at energies well above thresholds, but it is *not* always adequate and convergence tests should always be carried out. However, it can be drastically inefficient (and is unnecessary) for ultracold scattering including channels with $L > 0$, so the default value of IRXSET was changed from 1 to 0 from version 2019.0.

8.5.3 Propagator switch point R_{mid}

The LDJ, LDMD, LDMA and LDMG propagators have good step-size convergence in regions where the interaction potential is strong, but cannot take very long steps even when it is weak. The RMAT, VIVS and AIRY propagators have poorer step-size convergence when the interaction potential is strong, but can take much longer steps when it is weak. If different propagators or step-size algorithms are used at short and long range, R_{mid} should be chosen for optimum efficiency. It should usually be placed well outside the potential minimum, and it is often effective to place it between 75% and 99% of the way up the attractive limb of the potential.

For BOUND and FIELD, R_{mid} is set to RMID. If no value for RMID is provided, R_{mid} is set to RMATCH.

For MOLSCAT, if RVFAC = 0.0 (the default), R_{mid} is set to RMID.

If RVFAC > 0.0, R_{mid} is set to RVFAC $\times R_{\mathrm{turn}}$, where R_{turn} is an estimate of the position of the classical turning point in the lowest channel, calculated for each JTOT and symmetry block at the highest energy in ENERGY. Values of RVFAC from 1.3 to 2.0 are often satisfactory for cross-section calculations. If IRMSET = 0 and RVFAC > 0, R_{mid} is set to RVFAC $\times R_{\mathrm{min}}$.

RVIVAS is a deprecated synonym for RMID.

If $R_{\rm mid} < R_{\rm min}$, IPROPS is not used. If $R_{\rm mid} > R_{\rm max}$, IPROPL is not used.

8.5.4 Bound-state matching point $R_{\rm match}$

The value of $R_{\rm match}$ does not affect the energies or fields at which the matching condition (2.18) is satisfied, so it does not affect converged bound-state energies or fields. However, it does affect the log-derivative matching matrix at other energies or fields, so it can affect the rate of convergence on states (and sometimes the success of convergence).

A value of $R_{\rm match}$ somewhat inside the outer turning point (near the maximum in the outermost lobe of the wavefunction) is usually optimal for near-threshold bound states. For deeply bound states a value slightly outside the inner classical turning point usually gives rapid convergence.

It is usually inappropriate to place $R_{\rm match}$ far into a classically forbidden region, or at a distance where the log-derivative matrix has very large eigenvalues (such as very close to a hard wall in the interaction potential).

$R_{\rm match}$ is set to RMATCH. If no value for RMATCH is provided, $R_{\rm match}$ is set to RMID. At least one of RMID and RMATCH must be provided.

8.6 Step size

The programs offer three different approaches for choosing the propagation step size (length of propagation step). These are

- equally spaced steps;
- step size proportional to a power of R;
- adaptive step size based on error estimates.

Not all propagators implement all these approaches.

The step size(s) for the short-range propagator (IPROPS) are controlled by variables DRS or STEPS and EPS. Variable-step propagators use the additional variables TOLHIS and POWRS. The step size(s) for the long-range propagator (IPROPL) are controlled by corresponding variables DRL, STEPL, EPL, TOLHIL and POWRL.

DRL defaults to DRS.

STEPL defaults to STEPS.

EPS and EPL each default to 0.0, and setting one has no effect on the other.

TOLHIL defaults to TOLHIS.

POWRL defaults to $1.33\dot{3}$ if TOLHIL $= 0$ or 3.0 if TOLHIL > 0. POWRS defaults to 0.0 (equally spaced steps) except for the AIRY propagator (IPROPS $= 9$) with TOLHIL > 0, when it defaults to 3.0.

DR and TOLHI are deprecated synonyms for DRS and TOLHIS. POWRX is a deprecated synonym for POWRL.

In the remainder of Chapter 8, these variables are referred to without suffices as DR, STEP, EP, TOLHI and POWR.

The VIVS propagator has additional control variables that are not distinguished by suffices S and L, as described in section 8.8.3 below.

8.6.1 Equally spaced steps

All propagators except VIVS offer the option of equally spaced steps. For propagators where equally spaced steps are not the only option, they are selected by setting POWR = 0.0.

8.6.2 Step size proportional to $R^{\mathtt{POWR}}$

The RMAT and AIRY propagators allow a step size proportional to $R^{\mathtt{POWR}}$. POWR = 0.0 generates equally spaced steps. For the RMAT propagator, this mechanism is always used. For the AIRY propagator, it is used only if TOLHI = 0.0.

For POWR $\neq 1.0$, this option is implemented by choosing steps whose boundaries are equally spaced in the transformed variable $R^{1-\mathtt{POWR}}$. For the special case POWR = 1.0, it is implemented with the size of each step proportional to R at the *inner* end of the step.

Most systems of interest have interaction matrices that include centrifugal terms that are asymptotically diagonal in the adiabatic representation and decay as R^{-2} at long range, with first and second derivatives that decay as $W^{(1)} \propto R^{-3}$ and $W^{(2)} \propto R^{-4}$. For the RMAT propagator, the error in a single step is proportional to $W^{(1)}\delta R^3$ so the step size at long range should be proportional to R, which is achieved with POWR = 1.0. For the AIRY propagator the error is proportional to $W^{(2)}\delta R^3$, so POWR = $1.33\dot{3}$ is appropriate at long range.

Different values of POWR may be appropriate if a different inverse power is dominant. For the AIRY propagator, the error is proportional to $W^{(2)}\delta R^3$ for terms that are diagonal in the quasiadiabatic representation, but $W^{(1)}\delta R^3$ for terms that are off-diagonal. If the interaction potential decays as R^{-n}, it may be appropriate to use a step size proportional to $R^{(n+2)/3}$ if the R^{-n} potential terms are purely diagonal, or $R^{(n+1)/3}$ if they are off-diagonal. For example, for s-wave collisions between two atoms in S states, with diagonal potential terms that decay as R^{-6}, POWR = $8/3 = 2.66\dot{6}$ would be expected to offer optimum efficiency for the AIRY propagator at long range.

8.6.3 Adaptive step size

The VIVS and AIRY propagators can use adaptive step-size algorithms, with the size of each step based on an estimate of the errors in the previous step. The algorithms used are described in sections 8.8.3 and 8.8.5.

8.6.4 Initial step size

Setting initial step size with DRS or DRL

The default behaviour is to take the initial step size δR from DR. The input step size is modified slightly if necessary to give an integer number of steps over the propagation range.

If the step size is proportional to a power of R, DR is interpreted as the step size at the *inner* end of the range, even for inwards propagations. This allows the same value of DR to be used in BOUND, FIELD and MOLSCAT.

Setting initial step size with STEPS and EPS or STEPL and EPL

If STEP > 0.0, the step size is calculated from STEP and EP. This is interpreted as the number of steps per half-wavelength for the channel with the highest asymptotic kinetic energy $E_{\rm kin} = E - E_{{\rm intl},i}$. The step size is calculated from $\pi/(k \times \mathtt{STEP})$, where $k^2 = 2\mu(E_{\rm kin} + \mathtt{EP})/\hbar^2$. Thus EP may be input to estimate the depth of the interaction potential in the relevant region, to take account of the fact that the wavefunction varies faster over a potential well than asymptotically. EP is usually needed only if using the STEP mechanism at asymptotic kinetic energies smaller than the potential well depth. A value of STEP between 10 and 20 is usually adequate.

If ISCRU > 0, δR is calculated at the highest energy in ENERGY and this value is used for the propagations at all energies.

The value of δR obtained from STEP and EP is modified slightly if necessary to give an integer number of steps over the propagation range.

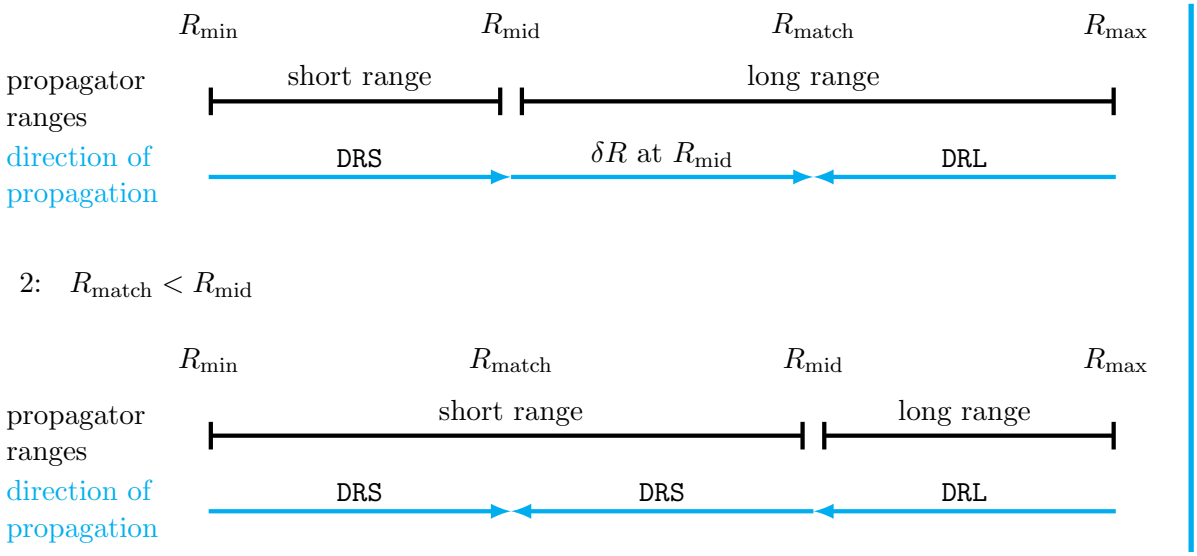
If the step size is proportional to a power or R, the step size obtained from STEP and EP is used at the *inner* end of the propagation range, even for inwards propagations. This allows the same values of STEP and EP to be used in BOUND, FIELD and MOLSCAT.

8.6.5 3-segment propagation in BOUND and FIELD

The wavefunction matching distance RMATCH may be different from the propagator switching point RMID, as described in section 8.5.4. If RMID and RMATCH are different, the programs perform a 3-segment propagation: outwards from RMIN to the smaller of RMID and RMATCH; inwards from RMAX to the larger of RMID and RMATCH; and from RMID to RMATCH (in whichever direction is needed).

If the propagation has three segments, there are two possibilities. The directions of propagation and the initial step sizes used are:

1: $R_{\rm mid} < R_{\rm match}$



2: $R_{\mathrm{match}} < R_{\mathrm{mid}}$

8.7 Convergence of propagations

It is very important to test convergence of coupled-channel calculations with respect to propagation range and step size. Lack of convergence can give very poor results, whereas unnecessarily conservative settings can waste large amounts of computer time. It is always advisable to conduct careful convergence tests (which can usually be done with a small basis set) before embarking on a major set of calculations. The programs provide automated mechanisms to facilitate this, described in section 9.5 for MOLSCAT and section 10.6 for BOUND.

8.8 Specific propagators

Most propagators are constructed to be exact for some *reference potential*, and incorporate deviations from the reference potential by approximation (or for some propagators not at all). Propagators that use a trivial reference potential (such as $\mathbf{W}(R) = \mathcal{E}\mathbf{I}$ in Eq. 2.6) are termed *solution-following* methods, whereas those that use a reference potential closer to the real potential, often without corrections, are termed *potential-following* methods.

The DV, LDJ and LDMG propagators are solution-following methods, while the RMAT and AIRY propagators are potential-following methods. The VIVS, LDMD and LDMA propagators are use intermediate schemes, incorporating both a reference potential and corrections to it. Solution-following methods must always take steps that are (much) smaller than the local wavelength. Potential-following and intermediate methods can sometimes be accurate even for step sizes larger than the wavelength, though the methods used to calculate node counts in BOUND and FIELD may fail if there is more than one node in any channel in a single step.

Propagators may operate either in the primitive basis set or in a quasiadiabatic representation, defined by diagonalising the interaction matrix $\mathbf{W}(R)$ at some point within the step. A quasiadiabatic representation is unchanged throughout the step, so should be distinguished

from a true adiabatic representation, which changes continuously with R; in a true adiabatic representation the coupled equations contain first-derivative coupling terms (involving d/dR rather than just d^2/dR^2), and such terms are not handled by the propagators in MOLSCAT, BOUND and FIELD. The DV, LDJ, LDMD and LDMG propagators operate in the primitive basis set, while the RMAT, VIVS, LDMA and AIRY propagators operate in a quasiadiabatic representation.

8.8.1 DV propagator (propagator 2)

The DV propagator uses the solution-following method of de Vogelaere [36] to propagate the wavefunction matrix $\mathbf{\Psi}(R)$ and its radial derivative $\mathbf{\Psi}'(R)$ in the primitive basis set. It allows only equally spaced steps.

The error in each step is proportional to δR^5, where δR is the step size. The total error after propagating across a range is proportional to δR^4.

The de Vogelaere method is potentially unstable for channels that are locally closed (i.e., in a classically forbidden region): the exponential growth of closed-channel wavefunctions can lead to a loss of linear independence of the solutions. To avoid this, the DV propagator re-imposes linear independence every `NSTAB` steps. The default is usually adequate.

8.8.2 RMAT propagator (propagator 3)

The RMAT propagator uses the potential-following method of Stechel *et al.* [37] to propagate the R matrix, which is the inverse of the log-derivative matrix.

The propagation is done in a quasiadiabatic basis set obtained by diagonalising the interaction matrix $\mathbf{W}(R)$ at the centre of each step. In each step, it uses a constant reference potential that is the interaction potential evaluated at the centre of the step in the quasiadiabatic basis.

The error in each step is proportional to δR^3, where δR is the step size. The total error after propagating across a range is proportional to δR^2. This is poorer than for the log-derivative methods and the RMAT propagator is recommended only for special purposes.

The RMAT propagator is implemented with equally spaced or power-law steps as described in sections 8.6.1 and 8.6.2. It is usually recommended to use it with equally spaced steps at short range and power-law steps at long range. It does not implement the adaptive step-size algorithm described in ref. [37].

8.8.3 VIVS propagator (propagator 4)

The VIVS propagator is an intermediate method that propagates the R matrix by the variable-interval variable-step method of Parker *et al.* [44]. It operates in a quasiadiabatic representation, with both a variable interval length and a variable step length. A single diagonalising transformation is used over the whole of each interval, which may consist of several steps. The error in each step or interval is proportional to δR^4, where δR is the step or interval size.

The VIVS propagator is designed for use at long range, but it usually offers poorer performance and stability than the AIRY propagator, and is recommended only for special purposes.

The step size and interval size algorithms used by VIVS attempt to take the longest step that gives the required accuracy at each point in the propagation. However, the optimum step size at one scattering energy is not necessarily safe at another, and VIVS can sometimes give inaccurate results at subsequent energies if the groups of energies in a particular run are not chosen with care. In particular, one should avoid:

1. A first energy that is close to (or above) a channel threshold and a subsequent energy that is far below it.
2. A first energy that is far above a channel threshold and a subsequent energy that is close to it.

Interval sizes

VIVS accumulates perturbation corrections to the wavefunction as it propagates, and uses these to obtain a suitable length for the next interval. The input parameter `DR` described above is used as the size of the first interval, and subsequent interval lengths are obtained using the input tolerance `TOLHI`; the criterion is that some functional t of the perturbation corrections should be not greater than `TOLHI` over any interval. Within an interval, t is tested against `TOLHI` at each step, and a new interval is started (with a new diagonalising transformation) if it appears likely to exceed `TOLHI` over the next step.

Even when the perturbation corrections are small, the algorithm used to obtain interval sizes limits the factor by which the interval size may be increased each time. The limit is more conservative in the presence of closed channels. In addition, when closed channels are present, the interval size is limited to $4/|k_n|$, where k_n is the local wavevector for the most deeply closed channel. Because of this, the VIVS propagator is not very efficient at long range in the presence of closed channels, and the AIRY propagator is more suitable in such cases.

Step sizes

Each interval is divided into `IALPHA` steps. Within an interval, the step sizes increase geometrically, with each step being a factor α larger than the previous one, subject to a maximum step size set by `DRMAX`. The quantity α may be specified in either of 2 ways:

1. If the logical input parameter `IALFP` is `.FALSE.`, α increases linearly from `ALPHA1` at the starting point for VIVS to `ALPHA2` at `RMAX`.
2. If `IALFP` is `.TRUE.` on input, the program starts with an initial value of `ALPHA1`, and adjusts this as it propagates. `ALPHA2` is then not used.

Automatic step and interval lengths

A useful special option is obtained by setting `IALPHA` $= 0$ on input. Intervals then consist of a variable number of steps, and the decision to start a new interval is based solely on the magnitude of the perturbation corrections; a new interval is started (and a new diagonalising

transformation obtained) whenever the quantity t approaches TOLHI. The initial step size is taken from DR, and subsequent steps use a criterion based on TOLHI.

The IALPHA = 0 option can be very efficient, and often requires remarkably few intervals/steps to produce converged results.

Perturbation corrections

There are several logical input variables that control the extent to which VIVS calculates and uses perturbation corrections to the wavefunction. The three variables IV, IVP and IVPP control the calculation of perturbation corrections due to the potential itself (IV) and to its first (IVP) and second (IVPP) derivatives. The perturbation corrections thus calculated are used in calculating interval sizes, but are included in the wavefunction only if IPERT is .TRUE.. If ISHIFT is .TRUE., the second derivative is used to shift the reference potential to give the best fit to the true potential.

For production runs, IV, IVP, IVPP, IPERT and ISHIFT should usually all be .TRUE. This may be forced by setting IDIAG = .TRUE., which overrides any .FALSE. values for the individual variables.

If IVP, IVPP or ISHIFT are .TRUE., VIVS requires radial derivatives of the interaction potential. These are supplied properly for simple potentials by the general-purpose version of POTENL described below, but for some potentials they can be difficult to evaluate. In this case, the input variable NUMDER may be set .TRUE., in which case the necessary derivatives are calculated numerically, and POTENL is never called with IC > 0; see section 16.1.2.

Other variables

ISYM If ISYM is .TRUE., the R matrix is forced to be symmetric at the end of each interval. This is usually advisable for production runs.

XSQMAX controls the application of perturbation corrections to deeply closed channels. If a channel is locally closed by more than XSQMAX reduced units, perturbation corrections for it are not calculated. The default should be adequate.

8.8.4 LDJ, LDMD, LDMA and LDMG propagators (propagators 5 to 8)

The LDJ [38, 39], LDMD [40], LDMA [41, 7] and LDMG [9] methods all propagate the log-derivative matrix $\mathbf{Y}(R) = \mathbf{\Psi}'(R)[\mathbf{\Psi}(R)]^{-1}$. Although $\mathbf{Y}(R)$ is discontinuous, the invariant-imbedding methods used to derive the propagators actually expand $\mathbf{\Psi}(R)$ and $\mathbf{\Psi}'(R)$, which are continuous functions.

The LDJ and LDMD propagators both operate in the primitive basis set. The LDJ propagator is a solution-following method, with the entire interaction matrix incorporated through quadrature. The LDMD propagator is an intermediate method that uses a reference potential that is the diagonal part of the interaction matrix $\mathbf{W}(R)$ at the centre of each step, and

treats only deviations from it by quadrature. The LDJ and LDMD propagators avoid the need to diagonalise $\mathbf{W}(R)$ at each step, but can be inefficient at long range if H_{intl} and/or $\hat{L}^2$ is non-diagonal.

The LDMA propagator is an intermediate method that uses a quasiadiabatic basis set defined by diagonalising $\mathbf{W}(R)$ at the centre of each step. Deviations from the reference potential are included by quadrature.

The LDJ, LDMD and LDMA propagators use a quadrature based on Simpson's rule to obtain an error in each step proportional to δR^5, where δR is the step size. This advantage applies only with an even number of steps, so internally the routines all propagate using half-steps of size $\delta R/2$. The total error after propagating across a range is proportional to δR^4.

The LDMG propagators are solution-following methods that take advantage of the symplectic nature of the multi-channel Schrödinger equation and reformulate it so that symplectic integrators (SIs) may be used to propagate solutions of the coupled equations. The variable IMGSEL specifies whether to use the five-step 4th-order method of Calvo and Sans-Serna (IMGSEL = 4) or the six-step 5th-order method of McLachlin and Atela (IMGSEL = 5). Both of these operate in the primitive basis set.

All these propagators operate only with equally spaced steps, and power-law steps are not currently implemented. For the LDJ, LDMD and LDMA propagators, a variable step size would lose the advantage of Simpson's rule and introduce errors proportional to a lower power of δR.

8.8.5 AIRY propagator (propagator 9)

The AIRY propagator uses the solution-following method of Alexander [42], as reformulated by Alexander and Manolopoulos [43]. It operates in a quasiadiabatic basis set defined by diagonalising $\mathbf{W}(R)$ at the central point R_i of each step i. It propagates the log-derivative matrix across the step using a linear reference potential $W_i(R) = W_i^{(0)} + W_i^{(1)}(R - R_i)$ in the quasiadiabatic representation.[†] Matrix elements off-diagonal in the quasiadiabatic basis set are neglected. The step size can increase rapidly with separation, so that this propagator is particularly efficient at long range.

DR is the size of the innermost step, except in the case of inwards propagation with TOLHI > 0, when it sets the size of the *first* step.

[†]The independent solutions of this piecewise linear potential are Airy functions of argument $W_i^{(0)}(W_i^{(1)})^{-2/3} + (W_i^{(1)})^{1/3}(R - R_i)$. For small arguments, the propagators are evaluated using the routines of Alexander and Manolopoulos [43]. However, as $W_i^{(1)}$ tends to zero at long range, the arguments of the Airy functions become asymptotically large; this can cause a loss of precision and may lead to numerical noise in derived quantities such as the eigenphase sum. Version 2019.0 and later implement a modified algorithm for evaluating the propagators at long range, using the expansions given in appendix A of ref. [52]; these are accurate as $W_i^{(1)} \to 0$. The asymptotic expansions are used if $\log(|W_i^{(0)}(W_i^{(1)})^{-2/3}|) >$ AC; the value of AC is hard-coded as 3.0 in subroutine SPROPN but can be changed for special purposes.

Adaptive step size

By default the AIRY propagator uses an adaptive step-size algorithm based on the variables TOLHI and POWR. TOLHI is a tolerance that is used to adjust the step size to try to maintain the same accuracy throughout the propagation. Values of TOLHI between 10^{-4} and 10^{-8} are generally useful.

The algorithm used to obtain step sizes for outwards propagations is that described by Alexander [42]. It calculates quantities that estimate two different sources of error:

CDIAG is proportional to the error due to neglected second-derivative terms in the potential that are diagonal in the local adiabatic basis set;

COFF is proportional to the error due to neglected first-derivative terms in the potential that are off-diagonal in the local adiabatic basis set.

The error from each source is proportional to the cube of the current step size, so the basic algorithm is for the step size in the next step to change from that in the current step by a factor $[\max\{\mathtt{CDIAG}, \mathtt{COFF}\}/\mathtt{TOLHI}]^{1/\mathtt{POWR}}$; POWR defaults to 3.0 when TOLHI > 0, but larger values may be input to limit the rate of increase/decrease. Any increase is limited to a factor of 2 in each step for stability.

The adaptive step-size algorithm is based on the expectation that the neglected derivatives of the potential are similar at the next step to those in the current step. For outwards propagation, the resulting step size generally increases with R at long range in a way that is sufficient to take account of the decreasing values of the neglected derivatives. However, for inwards propagation the derivatives are often larger at the next step than at the current one: for a potential R^{-n}, the first and second derivatives are proportional to R^{-n-1} and R^{-n-2}, and the basic algorithm might generate an over-long step. To maintain the error closer to a constant value, the predicted step size for inwards propagations is multiplied by an additional factor $(1-|\delta R|/R)^3$, which is sufficiently conservative to take account of potentials with inverse powers $n \leq 7$.

The AIRY propagator prints a warning if IPRINT ≥ 10 and the value of CDIAG or COFF in any step is greater than $5 \times$ TOLHI. This can occur if the step midpoint is close to a narrow avoided crossing between eigenvalues of $\mathbf{W}(R)$, since such avoided crossings produce narrow spikes in first-derivative terms that are off-diagonal in the local adiabatic basis set. However, such warnings can also occur if there is an unphysical discontinuity in the potential or its derivative, and this should be checked.

Noise due to adaptive step-size algorithm

The adaptive step-size algorithm can produce numerical noise in the values of R where the potential is evaluated, particularly when the range of eigenvalues of $\mathbf{W}(R)$ is large and part of the Hamiltonian (such as an external field) varies between calculations. This results in noise in S matrices and log-derivative matching matrices. The noise is usually small in absolute terms, but can impede convergence on resonances and bound states.

In BOUND, and when locating a scattering resonance as a function of energy in MOLSCAT (section 9.9), step-size noise can be eliminated by setting ISCRU > 0, so that calculations at

subsequent energies use exactly the same steps as at the first energy.

In FIELD, and when characterising a resonance in the scattering length as a function of EFV in MOLSCAT (section 9.10.3), step-size noise can be eliminated by setting TOLHI = 0 to choose equally spaced or power-law step sizes, as described in sections 8.6.1 and 8.6.2, in place of the adaptive step-size algorithm with TOLHI > 0 (but see warnings below).

Choice of adaptive or power-law step sizes

The adaptive step-size algorithm is easy to control and robust for outwards propagation. POWR should almost always be set to 3.0, and the accuracy is controlled by the single variable TOLHI. If the step size is initially too small, the algorithm quickly increases it without much inefficiency. For outwards propagations, it allows the step size to increase very fast in regions where the error is dominated by potential terms proportional to R^{-n} with $n > 2$, and then moderates the rate of increase when centrifugal terms become dominant. It usually takes very long steps at long range, so that using very large values of RMAX is inexpensive.

The adaptive step-size algorithm is less robust for inwards propagation, as often chosen for BOUND and FIELD at long range. When some channels have $L > 0$, the curvature of the centrifugal potential usually prevents excessive step sizes and the algorithm works well. However, the case where all channels have $L = 0$ is problematic (and occurs, for example, for atomic collisions with $L_{\max} = 0$). In this case the step size can grow very large, and not decrease fast enough to provide sufficient points at shorter range. For inwards propagation in cases where $L = 0$ for all channels, step sizes proportional to $R^{\mathtt{POWR}}$ are recommended; for potentials that decay as $1/R^6$, as for neutral atoms, POWR = $2.666\dot{}$ is often appropriate.

Step sizes proportional to $R^{\mathtt{POWR}}$ eliminate step-size noise, but take more expertise to specify and are often less efficient for a given accuracy. In this case, DR must be chosen with care, since the step size is proportional to it throughout the range; it is very inefficient to use too small a value. A single value of POWR must be used throughout the range, and it must be chosen conservatively, usually as POWR = $1.333\dot{}$ to accommodate centrifugal terms as described in section 8.6.2; this may mean forgoing fast increases in step size achieved by the adaptive algorithm at shorter range. Finally, the values of DR and POWR needed at shorter range may result in much smaller steps at long range than are achieved by the adaptive algorithm, and this in turn may require greater care in choosing a value of RMAX that is not unnecessarily large.

For these reasons, we recommend using the AIRY propagator with the adaptive step-size algorithm (TOLHI > 0) in the first instance, and switching to power-law steps (TOLHI = 0) only if convergence difficulties due to step-size noise are encountered, with careful evaluation of the different values that may be needed for DR, POWR and RMAX.

8.8.6 WKB integration (propagator -1)

WKB integration is not strictly a propagator and is suitable only for single-channel cases (particularly in IOS calculations). It evaluates the WKB integral for the phase shift by Gauss-Mehler quadrature. It may not be combined with any other propagator. It is controlled by

input variables NGMP and TOLHI.

NGMP Dimension 3. N-point Gaussian integration is performed starting with $N =$ NGMP(1), incrementing by NGMP(2), until NGMP(3). Starting with the 2nd pass, the phase shift is compared with the previously calculated value until it has converged to within a tolerance specified by TOLHI.

TOLHI is the convergence tolerance for the WKB phase shift.

8.9 Adiabats and nonadiabatic matrix elements

The LDMA propagator includes features to print adiabats and nonadiabatic matrix elements between the adiabatic functions. These are often useful in interpreting bound states and scattering.

The adiabats $U_i(R)$ are the eigenvalues of

$$H_{\text{intl}} + V(R, \xi_{\text{intl}}) + \frac{\hbar^2 \hat{L}^2}{2\mu R^2} \tag{8.2}$$

at fixed values of R. They are simply the eigenvalues of the interaction matrix $\mathbf{W}(R)$, rescaled into energy units. The adiabatic functions $\Phi_i^{\text{ad}}(\xi_{\text{intl}}; R)$ are obtained from the corresponding eigenvectors; the semicolon indicates that the functions depend only parametrically on R.

The nonadiabatic matrix elements $B_{ij}(R)$ are

$$B_{ij}(R) = \int \Phi_i^{\text{ad}*}(\xi_{\text{intl}}; R) \frac{d}{dR} \Phi_j^{\text{ad}}(\xi_{\text{intl}}; R)\, d\xi_{\text{intl}}. \tag{8.3}$$

The diagonal elements $B_{ii}(R)$ are zero and the off-diagonal elements are evaluated using the Hellmann-Feynman theorem [53],

$$B_{ij}(R) = \frac{\displaystyle\int \Phi_i^{\text{ad}*}(\xi_{\text{intl}}; R) \left[\frac{d}{dR} \left(V(R, \xi_{\text{intl}}) + \frac{\hbar^2 \hat{L}^2}{2\mu R^2} \right) \right] \Phi_j^{\text{ad}}(\xi_{\text{intl}}; R)\, d\xi_{\text{intl}}}{U_j(R) - U_i(R)}. \tag{8.4}$$

At the midpoint of each step, the program evaluates the matrix $d\mathbf{W}/dR$, transforms it into the adiabatic basis set using the eigenvectors of $\mathbf{W}(R)$, and then divides by the appropriate denominators to obtain $B_{ij}(R)$. It should be noted that the nonadiabatic matrix elements peak sharply at values of R that correspond to narrow avoided crossings between adiabats, where $U_j(R) - U_i(R)$ can be very small. The absolute signs of the eigenvectors of $\mathbf{W}(R)$ are numerically arbitrary and may change between propagation steps; as a result, the signs of the matrix elements produced from Eq. 8.4 may also change sign arbitrarily.

There are also both diagonal and off-diagonal matrix elements of the operator d^2/dR^2 as described in ref. [53]. These are not currently evaluated but would be straightforward to add.

The print levels needed for these features are described in section 12.5.

8.10 Boundary conditions

All the programs carry out outwards propagations starting at $R_{\rm min}$, and BOUND and FIELD also carry out inward propagations starting at $R_{\rm max}$. Boundary conditions are needed to initialise these propagations. Versions of the programs before 2019.0 used a variety of boundary conditions, which were adequate for most cases, but better control is needed for special purposes. From version 2019.0, the choice of boundary conditions for log-derivative propagators has been unified, and variables ADIAMN, ADIAMX, BCYCMN, BCYCMX, BCYOMN, BCYOMX, WKBMN and WKBMX have been introduced to control them. The variables ...MX are not used in MOLSCAT.

The boundary condition is always expressed as a diagonal log-derivative matrix, but it may be diagonal either in the primitive basis set or in the locally adiabatic representation that diagonalises the coupling matrix $\boldsymbol{W}(R)$. In versions of the programs before 2019.0, the boundary conditions were applied in the primitive basis set for diabatic propagators and in the adiabatic basis set for adiabatic propagators. From version 2019.0 the choice is controlled by the namelist items ADIAMN and ADIAMX: these are logical variables (default .TRUE.) that specify whether the boundary conditions at $R_{\rm min}$ and $R_{\rm max}$, respectively, are applied in the adiabatic basis. If necessary, the resulting diagonal log-derivative matrix is transformed into the basis set used by the propagator chosen.

For a single channel, the log-derivative is defined as $Y = \psi'(R)/\psi(R)$. At the wall of a box, $\psi(R_{\rm wall}) = 0$, so $Y = \infty$. However, it is sometimes useful to define a boundary condition corresponding to $\psi'(R) = 0$ and $\psi(R)$ finite, so that $Y = 0$. Lastly, if the wavefunction follows the WKB approximation in the classically forbidden region, then

$$\psi(R) = [k(R)]^{-\frac{1}{2}} \exp\left(\pm \int_{R_{\rm turn}}^{R} k(R')\, dR'\right), \tag{8.5}$$

$$\psi'(R) = [k(R)]^{-\frac{1}{2}} \left[\pm k(R) - \frac{1}{2}\frac{k'(R)}{k(R)}\right] \exp\left(\pm \int_{R_{\rm turn}}^{R} k(R')\, dR'\right), \tag{8.6}$$

$$Y(R) = \pm k(R) - \frac{1}{2}\frac{k'(R)}{k(R)}, \tag{8.7}$$

where $k(R) = [2\mu(V(R) - E)/\hbar^2]^{1/2}$ and $V(R)$ is an effective potential energy for the channel concerned. The + sign applies inside the inner turning point (where the phase integral is itself negative) and the − sign applies outside the outer turning point. The first term in Eq. 8.7 dominates either when $k(R)$ is large (in a strongly classically forbidden region) or when the interaction potential is nearly constant (at very long range). The term involving $k'(R)$ is therefore neglected in the implementation of WKB boundary conditions.

Locally closed channels

For locally closed channels, the default is to use WKB boundary conditions at both $R_{\rm min}$ and $R_{\rm max}$ (WKBMN = WKBMX = .TRUE.). WKB boundary conditions usually give the fastest convergence with respect to $R_{\rm min}$ and $R_{\rm max}$ for problems where the interaction potential remains finite in the classically forbidden region. For special purposes, WKBMN and/or WKBMX may be set .FALSE. and explicit values provided in the variables BCYCMN and/or BCYCMX. These values are used for all locally closed channels at $R_{\rm min}$ and $R_{\rm max}$, respectively.

Locally open channels at $R_{\rm min}$

It is relatively rare to start a propagation in the presence of locally open channels at $R_{\rm min}$. By default, BCYOMN is unset and the programs stop if they detect locally open channels at $R_{\rm min}$. If locally open channels actually exist, BCYOMN must be set explicitly.

For systems that have no classically forbidden region at short range, it may be appropriate to start the propagation at $R = 0$. When R is a true radial coordinate with volume element R^k and $k > 0$, $Y(0)$ is usually ∞ and BCYOMN should be set to a large positive value (e.g., 10^8) to represent this as described below. For 1-dimensional systems where R does not represent a radial coordinate, states that are symmetric about $R = 0$ require $Y(0) = 0$ and those that are antisymmetric require $Y(0) = \infty$.

Locally open channels at $R_{\rm max}$

It is more common to start an inwards propagation in the presence of locally open channels at $R_{\rm max}$. It often occurs, for example, when running FIELD at the energy of a scattering threshold to locate the fields at which bound states cross the threshold. It is also sometimes desired to use BOUND or FIELD to estimate the positions of quasibound states that lie above threshold, applying a boundary condition with BCYOMX at long range to quantise the open channels. By default, BCYOMX = 0.0. The reason for this choice is that the WKB boundary condition (8.7) implies that $Y(R_{\rm max}) \to 0$ as the energy E approaches $V(R_{\rm max})$ from below. The choice BCYOMX = 0.0 provides continuity across this energy when WKB boundary conditions are used for locally closed channels. Discontinuities in boundary conditions may cause discontinuities in the node count and disrupt convergence on bound-state positions.

Finite values representing infinity

There are several situations where the physical boundary condition required is $Y(R) = \infty$. These include an infinite hard wall and the behaviour at the origin in polar or spherical polar coordinates. Such boundary conditions should be specified with large positive values (e.g., 10^8) for BCYCMN or BCYOMN and with large negative values (e.g., -10^8) for BCYCMX or BCYOMX. Using large values with signs different from these has little effect on scattering properties or bound-state positions, but in BOUND and FIELD it may affect the node count, as it causes an extra node to appear very close to the end-point in each channel.

Values larger than 10^8 may be used if required, but can sometimes cause problems if the initialisation is done in one representation (adiabatic or diabatic) and then transformed to the other. This is particularly true if different boundary conditions are used for different channels (e.g., WKB boundary conditions for locally closed channels and large values for locally open channels). In such circumstances, initialisation should be done in the same representation as the propagation.

8.11 Propagator scratch file

All the propagators have options to save energy-independent matrices on a scratch file to save computation at subsequent energies. If ISCRU > 0, this save file is created on unit ISCRU and used for any subsequent energies. It can be large.

This option saves CPU time at the expense of disc I/O. It is often advantageous for the LDMD, LDMA, AIRY, RMAT and VIVS propagators if NNRG is not 1, but for the DV and

LDJ propagators it does not usually save resources overall unless the interaction potential itself is very expensive to evaluate. This is particularly true on machines where the discs are accessed over a network. Note, however, that ISCRU > 0 may save considerable time for single-channel IOS cases with the LDMD propagator, since the computer time for these is often dominated by potential evaluations.

For the AIRY propagator, the option ISCRU > 0 also ensures that the steps taken by the adaptive step-size algorithm are identical at subsequent energies to those at the first energy. This may be useful in reducing step-size noise, as described on p. 85.

For the special case of MOLSCAT calculations for a single value of `JTOT` and `IBLOCK` and a single set of EFVs, the `ISCRU` file from one run may be used as input for the next run at a different set of energies. The routine `mol.driver.f` must first be recompiled with a different `OPEN` statement for unit `ISCRU`, which must omit the specification `STATUS='SCRATCH'`. The relevant lines of code are currently commented out in `mol.driver.f`. The first run then produces a file on unit `ISCRU` that is preserved. In subsequent runs, ISCRU should be set negative; the program then expects to find the file from the first run on unit |ISCRU|. It reads the header on this file to check that it contains valid information, and then proceeds with "subsequent energy" calculations for all the energies requested.

Chapter 9

Controlling scattering calculations

9.1 Asymptotic basis sets

If the operators H_{intl} and $\hat{L}^2$ are diagonal in the basis set used for the propagation, the propagated wavefunction matrix $\Psi(R_{\mathrm{max}})$ and its derivative $\Psi'(R_{\mathrm{max}})$, or the log-derivative matrix $\mathbf{Y}(R_{\mathrm{max}})$, are matched directly to analytic radial functions that describe the solutions of the Schrödinger equation in the absence of an interaction potential to obtain the K matrix. This is then converted into the scattering S matrix, as described in section 2.6. This procedure is used for all the built-in interaction types and for some plug-in basis-set suites.

If one or both of H_{intl} and $\hat{L}^2$ are not diagonal in the basis set used for the propagation, the propagated functions are transformed into an asymptotic basis set that diagonalises them. In the simplest (and most common) case, MOLSCAT constructs the matrices of H_{intl} and/or $\hat{L}^2$ as necessary, diagonalises them, and transforms the wavefunction or log-derivative matrices before matching to analytic radial functions to obtain the K and S matrices. The rows and columns of the K and S matrices are labelled by the asymptotic basis functions, but these functions are not generally described in any simple way by the quantum numbers in the array `JSTATE`.

9.2 Resolving degeneracies with extra operators

In some cases, the requirement that the asymptotic basis functions are eigenfunctions of H_{intl} and $\hat{L}^2$ is not enough to define them uniquely. This occurs if two or more channels with the same value of L are degenerate in energy (or very nearly degenerate). Numerical diagonalisation of the matrix $\boldsymbol{H}_{\mathrm{intl}}$ may then produce eigenvectors that are linear combinations of the physically relevant vectors. Plug-in basis-set suites may be programmed to specify extra operators $\hat{P}_i$ to aid in resolving such (near-)degeneracies. These extra operators, which need not necessarily contribute to H_{intl} or $\hat{L}^2$, must nevertheless commute with them. In any (near-)degenerate subspace of the eigenvectors of H_{intl} and $\hat{L}^2$, MOLSCAT constructs the matrix of the first such operator ($\hat{P}_1$) and finds linear combinations of the degenerate functions that are eigenfunctions of it. If the eigenvalues of $\hat{P}_1$ are sufficiently non-degenerate, the process

ends; if not, it is repeated with operator $\hat{P}_2$, and so on.

The namelist item DEGTOL is used as a threshold for degeneracy for both $H_{\rm intl}$ and the extra operators. It is treated as an energy and so is scaled according to EUNITS or EUNIT.

9.3 Channel indices for open channels

The S matrix has dimension $N_{\rm open} \times N_{\rm open}$, where $N_{\rm open}$ is the number of open channels. The open channels for each propagation are sorted in order of increasing threshold energy, and MOLSCAT prints a list of them after each propagation if IPRINT ≥ 10.

The list of open channels gives the open-channel index and the index of the corresponding channel in the complete channel list (described in sections 4.2.1 and 4.2.2). It also repeats the associated value of L, the index of the pair level and the corresponding pair energy (threshold energy). If IBOUND $= 1$, the diagonal matrix element $\langle \hat{L}^2 \rangle$ stored in the CENT array is printed in place of the integer L.

9.4 S matrix

If IPRINT ≥ 11, MOLSCAT prints the S matrix in the main output file. Each element is labelled by the initial and final open-channel indices. The output gives the square modulus, the phase, and the real and imaginary parts. Only elements with square modulus greater than 10^{-20} are printed.

If the S matrix is to be read by an external program, it is usually more convenient to obtain it from an auxiliary output file written on channel ISAVEU, as described in section 13.5, rather than from the main output file.

9.5 Automated testing of propagator convergence

If NCONV > 0, MOLSCAT performs NCONV extra calculations of the S matrix, with different values of RMIN, RMAX or the step size, and compares the results. Which one of these lengths is varied is governed by ICON as follows:

ICON $= 1$ doubles the initial step size each time
ICON $= 2$ decreases RMAX by DRCON each time
ICON $= 3$ increases RMIN by DRCON each time

MOLSCAT calculates an S matrix and prints the root-mean-square change in S-matrix elements and transition probabilities each time. This provides an automated means of choosing propagation parameters capable of providing the required accuracy. Only a single S matrix is stored between calculations, so the program must not loop over angular momenta, symmetry blocks, energies, or sets of EFVs; thus JTOTU must be equal to JTOTL, IBFIX must be set (and IBHI must be unset), NNRG must be 1 and NFIELD must be 1. In addition, ISCRU must be 0.

If ICONVU $= 0$ (the default), the results from each propagation are compared with the first. If

ICONVU > 0 on input, the first S matrix is written (unformatted) to unit ICONVU; if ICONVU < 0, a previously saved S matrix is read from unit |ICONVU|, and used as the reference S matrix in calculating root-mean-square (rms) errors.

Remember that, in some modes, MOLSCAT determines RMIN, RMID and RMAX internally, and it is safest to test convergence with these options switched off: IRMSET $= 0$, RVFAC $= 0.0$, IRXSET $= 0$.

9.6 Elastic and inelastic cross sections

For the built-in interaction types, MOLSCAT calculates degeneracy-averaged elastic and inelastic cross sections (and contributions to them from each JTOT and IBLOCK) between the levels in the JLEVEL array (see Eq. 2.12). Cross sections are labelled by the indices of initial and final levels. These are given as PAIR LEVEL in the list of pair states that is printed if IPRINT ≥ 1. It is possible to limit the calculation to a subset of the levels using MXSIG, which also serves to control how much is printed.

The same structures are used for plug-in basis-set routines that implement basis sets in which $H_{\rm intl}$ and $\hat{L}^2$ are diagonal.

If one or both of $H_{\rm intl}$ and $\hat{L}^2$ is non-diagonal, the asymptotic energy levels for a particular symmetry block are not known until the basis set for that symmetry block is constructed and cannot be described by JLEVEL. In this case, MOLSCAT analyses the array of eigenvalues of $H_{\rm intl}$ from successive values of JTOT and IBLOCK and constructs a master array of channel threshold energies in the array ELEVEL. Channels that have energies within a convergence criterion DEGTOL are assumed to originate from the same pair level. MOLSCAT assigns a level index to each open channel, stored in the array INDLEV, and (if requested) accumulates cross sections between the stored levels. If IPRINT ≥ 10, MOLSCAT prints the assignment of channels to levels for each JTOT/IBLOCK combination. After the loops over JTOT and/or IBLOCK, MOLSCAT has a complete list of levels that are open at one or more collision energies. It prints this list, with the corresponding energies, immediately before the degeneracy-averaged cross sections.

For basis sets in which $H_{\rm intl}$ is non-diagonal, MOLSCAT has no knowledge of any monomer quantum numbers associated with individual levels. The levels must be identified on physical grounds. If necessary, the eigenvectors that connect the asymptotic channels to the primitive basis functions (for each JTOT and IBLOCK) may be printed as described in section 4.2.2.

This scheme for identifying levels is compatible with calculations at multiple collision energies, but *not* with calculations at multiple values of external fields, since threshold energies depend on external fields. MOLSCAT calculates cross sections only if there is just one set of EFVs in a run*. Because the number of pair energy levels may not be limited by NLEVEL, the user must give MXSIG a positive value to limit the number of pair energy levels between which cross sections are calculated. If the reference energy is obtained from IREF, cross sections are calculated only in runs that are limited to a single value of JTOT and a single symmetry block.

*MOLSCAT can, however, calculate cross sections for scans across the potential scaling factor

9.6.1 Partial cross sections

The contributions to state-to-state cross sections from individual values of JTOT and IBLOCK (partial cross sections) are calculated after each propagation and printed if IPRINT ≥ 5. There was previously an option to print them out to unit IPARTU if IPARTU > 0; this option is currently disabled, but would not be difficult to revive if needed.

9.6.2 Restarting runs to calculate cross sections

If a particular calculation terminates prematurely (e.g., crashes or runs out of time), but the S matrices have been saved on ISAVEU, it is possible to restart the run from midway through the sequence of propagations. If IRSTRT > 0, MOLSCAT checks that the results stored on ISAVEU are for the same calculation and recalculates cross sections from S matrices included in the file. It then continues with subsequent propagations and S matrices.

- If IRSTRT $= 3$, the program restarts after the last complete propagation;
- If IRSTRT $= 2$, the program restarts after the last complete symmetry block;
- If IRSTRT $= 1$, the program restarts after the last completed value of JTOT.
- If IRSTRT $= -1$, the program extends the set of values for JTOT upwards to a new (and larger) value of JTOTU.

9.6.3 Integral cross sections

The state-to-state partial cross sections are accumulated to form the state-to-state integral cross sections for each collision energy. By default, MOLSCAT reserves storage for, and accumulates, integral cross sections between every pair of levels included in the calculation. This may use a substantial amount of storage, and is often not required, particularly if some of the levels are energetically inaccessible at all collision energies. If MXSIG > 0, only cross sections between the first MXSIG levels are calculated.

If ISIGU > 0, a direct access-file is opened on unit ISIGU and is used to store the accumulated state-to-state cross sections. It is updated after each propagation, so contains useful information on the calculation so far, even if the program terminates abnormally.

Automated convergence with respect to JTOT

Integral cross sections are calculated by accumulating cross sections for all possible values of JTOT from JTOTL to JTOTU in steps of JSTEP. However, if JTOTU ≥ 99999 (or JTOTU $<$ JTOTL) (the default)), the loop terminates when contributions from successive values of JTOT are negligible. Termination of the loop is controlled by the variables DTOL, OTOL and NCAC: JTOT starts at JTOTL and is incremented by JSTEP until NCAC successive values of JTOT each contribute less that DTOL to any diagonal cross section and less than OTOL to any off-diagonal cross section.

The final cross sections are multiplied by JSTEP to account (approximately) for incomplete sampling of JTOT values, unless JHALF $= 0$, indicating that JTOT is not used for J_{tot}.

This option should be used with care to ensure that the calculation converges before the job runs out of time. Note that elastic cross sections usually converge very much more slowly than inelastic ones.

It is sometimes desired to test the convergence of cross sections with respect to JTOT for a group of energies together, rather than one energy at a time. This option is controlled by the namelist item NNRGPG, which specifies the number of (successive) energies to be considered together.

9.7 Line-shape cross sections

For many of the built-in interaction types, MOLSCAT can calculate the cross sections that characterise pressure broadening, pressure shifting and pressure-induced mixing of spectroscopic lines. Each cross section is labelled by a pair of spectroscopic lines, $n_\mathrm{a} \to n_\mathrm{b}$ and $n'_\mathrm{a} \to n'_\mathrm{b}$. The real and imaginary parts of diagonal cross sections ($n_\mathrm{a} = n'_\mathrm{a}$ and $n_\mathrm{b} = n'_\mathrm{b}$) describe pressure broadening and shifting (respectively) of an isolated line, while the off-diagonal matrix elements describe line mixing.

As an example, the line-shape cross sections for close-coupling calculations on atom + vibrating diatom collisions, where n represents vibrational and rotational quantum numbers v and j, are [19]

$$\sigma(v_\mathrm{a} j_\mathrm{a}, v_\mathrm{b} j_\mathrm{b} | v'_\mathrm{a} j'_\mathrm{a}, v'_\mathrm{b} j'_\mathrm{b}; E_\mathrm{kin}) = \frac{\pi}{k_j^2} \sum_{\substack{LL' \\ J_\mathrm{a} J_\mathrm{b}}} \begin{Bmatrix} j_\mathrm{a} & q & j_\mathrm{b} \\ J_\mathrm{b} & L & J_\mathrm{a} \end{Bmatrix} \begin{Bmatrix} j'_\mathrm{a} & q & j'_\mathrm{b} \\ J_\mathrm{b} & L' & J_\mathrm{a} \end{Bmatrix} \times \left[I - S^{J_\mathrm{b}}_{i_\mathrm{b} f_\mathrm{b}}(E^\mathrm{kin}_\mathrm{b})^* S^{J_\mathrm{a}}_{i_\mathrm{a} f_\mathrm{a}}(E^\mathrm{kin}_\mathrm{a}) \right], \tag{9.1}$$

where $E^\mathrm{kin}_\mathrm{x} = E_\mathrm{kin} + E_\mathrm{x}$ and I is shorthand for $\delta_{v_\mathrm{a} v'_\mathrm{a}} \delta_{v_\mathrm{b} v'_\mathrm{b}} \delta_{j_\mathrm{a} j'_\mathrm{a}} \delta_{j_\mathrm{b} j'_\mathrm{b}} \delta_{L,L'}$. q is the tensor order of the spectroscopic transition, and unprimed and primed quantities refer to values before and after a collision. It should be noted that the two S matrices involved here are evaluated at the same *kinetic* energy E_kin but different *total* energies.

Calculations of line-shape cross sections are implemented for the ITYPEs indicated with reference numbers or black tick marks as follows.

		CC (+0)	EP (+10)	CS (+20)	DLD (+30)	IOS (+100)
rigid rotor + atom	1	[54]	[55]	[56]	[57]	[58]
vibrating rotor + atom	2	[54, 59]	✓	✓		
rigid rotor + rigid rotor	3	[60]				
asymmetric top + rigid rotor	4					
symmetric top + atom	5	[24]	✓	[24]		[61]
asymmetric top + atom	6	[24]	✓	[24]		
vibrating rotor + atom	7	[54, 27]	✓	✓		
atom + corrugated surface	8	N/A	N/A	N/A	N/A	N/A

Line-shape cross sections are calculated if NLPRBR > 0 on input; the value of NLPRBR specifies the number of (pairs of) spectroscopic lines for which line-shape calculations are required.

The lines themselves are specified by the array LINE, of $4 \times$ NLPRBR elements. Each successive quartet of elements in LINE specifies the two pairs of levels involved in the two transitions (n_a, n_b, n'_a, n'_b) as pointers to the JLEVEL and ELEVEL array; see section 4) for details.

Line-shape cross sections require S-matrix elements involving the initial and final (spectroscopic) levels at the same kinetic (not total) energy. If IFEGEN > 0, the program treats the input ENERGY values as kinetic energies ($E_{\rm kin}$ in Eq. 9.1), and generates the necessary total energies for the lines requested (i.e., $E_a^{\rm kin}$ and $E_b^{\rm kin}$ in Eq. 9.1). If IFEGEN $= 0$, only those requested lines for which cross sections can be constructed from the total energies actually specified by NNRG and ENERGY are calculated. Only total energies needed for the requested line-shape calculations are retained. In addition, specifying IFEGEN > 1 suppresses calculations for individual combinations of JTOT, IBLOCK and energy that do not contribute S matrices needed for the requested line-shape cross sections. *Warning:* some of the state-to-state integral cross sections may be incomplete (missing contributions from some values of JTOT and IBLOCK) when IFEGEN > 1.

The tensor order of the spectroscopic transition (i.e., 1 for dipole transitions and 0 or 2 for isotropic or anisotropic Raman scattering respectively) is specified in the input array LTYPE. If the default value is found, LTYPE is calculated as the difference between the rotational quantum numbers of the levels specified (taken from the JLEVEL array). The default is usually adequate *except* for Q-branch lines in ITYP $= 5$ or 6 or anisotropic Raman spectra with $\Delta j \neq 2$.

9.8 Line-shape cross sections in the IOS approximation

This code is at present unsupported, but is left in case someone wants to develop it.

9.9 Locating scattering resonances as a function of energy

This section is for resonances that appear in the energy dependence of S matrices. Resonances that appear as a function of external field at constant kinetic energy, such as magnetically tunable Feshbach resonances in low-energy collisions, should be located as described in section 9.10.3.

These options are not supported for IOS calculations (ITYPE > 100).

Scattering resonances and predissociating states of Van der Waals molecules appear as characteristic features in the energy dependence of S matrices. The eigenphase sum $\mathcal{S}$, which is the sum of phases of the eigenvalues of the S matrix [62], follows a Breit-Wigner form in the vicinity of a resonance,

$$\mathcal{S}(E) = \mathcal{S}_{\rm bg}(E) + \arctan\left(\frac{\Gamma}{2(E_{\rm res} - E)}\right), \tag{9.2}$$

where $E_{\rm res}$ is the energy of the resonance, Γ is its width, and $\mathcal{S}_{\rm bg}(E)$ is a slowly varying background phase. If IPHSUM > 0, the eigenphase sum is calculated and a summary of the eigenphases is output on unit IPHSUM.

The product state distribution from decay of a quasibound state is characterized by a set of partial widths Γ_i for each open channel i. For an isolated narrow resonance, the partial widths sum to the total width Γ. Across a resonance, each S-matrix element describes a circle in the complex plane [63, 64],

$$S_{ii'}(E) = S_{\mathrm{bg},ii'}(E) - \frac{\mathrm{i}g_i g_{i'}}{E - E_{\mathrm{res}} + \mathrm{i}\Gamma/2}. \tag{9.3}$$

The partial widths are defined as real quantities, $\Gamma_i = |g_i|^2$, and the circles in the complex plane have radii $\sqrt{\Gamma_i \Gamma_{i'}}/\Gamma$.

In cases where the background phase does not vary significantly across the width of the resonance, and the location of the resonance is approximately known in advance, MOLSCAT can converge on the resonance and obtain its position, width and background phase using the algorithm of Frye and Hutson [12]. If the energy dependence of the background phase can be estimated independently, a user-supplied routine `BCKGRD` may be provided to subtract it from the calculated phase.

If `IECONV` = 4, MOLSCAT performs scattering calculations at the first 3 energies specified as described in Chapter 6. It then uses the algorithm of Frye and Hutson [12] to attempt to converge on and characterise the resonance. The variables `TLO`, `THI` and `XI` set values for the parameters t_{lo}, t_{hi} and ξ of ref. [12]. The characterisation terminates when one of the three points is within `DTOL` of the estimated value of E_{res} and the other two are within the ranges prescribed by `TLO`, `THI` and `XI`. It also ends if `MXLOC` (set in module `sizes`) propagations are performed without convergence. The current estimates of the resonance parameters are printed at each step.

If `IECONV` = 5, MOLSCAT uses an algorithm based on the fully complex procedure of ref. [13] to attempt to converge on a resonance in a single diagonal S-matrix element and extract parameters E_{res}, Γ, g_i^2 and $S_{\mathrm{bg},ii}$. The channel concerned is identified by its open-channel index `ICHAN` (section 9.3). The sequence of points is controlled by `TLO`, `THI` and `XI` in the same way as for `IECONV` = 4.

The default values `TLO` = −0.1 and `THI` = 1.0 are usually appropriate for narrow resonances. However, larger values of t_{lo} and t_{hi} are sometimes needed for extremely narrow resonances, to reduce the effects of numerical noise, and smaller values may be needed for very wide resonances, to reduce variation in the background across the range. The default value `XI` = 0.25 is usually appropriate for general resonance characterisation, but much smaller values may be needed for special purposes (such as least-squares fitting to determine interaction potentials) to avoid small discontinuities in calculated resonance properties as a function of potential parameters.

If convergence succeeds when `IECONV` is either 4 or 5, MOLSCAT also calculates and prints partial widths based on the two final points closest to and furthest from E_{res}.

To handle cases where the automated algorithm is unsatisfactory, MOLSCAT outputs K matrices instead of S matrices on channel `ISAVEU` when `IPHSUM` > 0. If the run is for a single value of `JTOT` and a single symmetry block, the K-matrix file is suitable for input to the separate program `SAVER`, which can accumulate results from several different runs. The accumulated K matrices may then be processed by external program `RESFIT` [4] to obtain resonance positions,

widths and partial widths.

Broad resonances may often be identified from features in cross sections (usually peaks, but sometimes troughs or more complicated features). However, very narrow resonances (corresponding to long-lived quasibound states) can be hard to find. This is compounded by the fact that, although the eigenphase sum $\mathcal{S}$ increases smoothly by π across the width of a resonance, it is numerically defined only modulo π when evaluated from a K or S matrix. MOLSCAT chooses the integer part of $\mathcal{S}/\pi$ so that it is around at 10 at the first energy, and then chooses the integer part at each subsequent energy to be within 0.5 of the value at the previous energy. Thus, if successive energies fall more than about $\Gamma/5$ below and above a resonance, the resonance may appear as an irregularity in $\mathcal{S}/\pi$, rather than a smooth increase through 1. If the spacing is much larger than Γ, the presence of the resonance may be hard to spot.

MOLSCAT has a capability to identify a resonance from calculations in its wings, and to step towards it when the *curvature* of the eigenphase sum is dominated by the resonant contribution. This may succeed in locating a resonance from further away than the automated algorithm of ref. [12], particularly in the presence of an unknown background slope. If IECONV $= -5$ or NNRG < 0, MOLSCAT generates 5 initial energies from ENERGY(1) and DNRG and performs n groups of 5 equally spaced calculations, where n is the integer part of |NNRG|/5. After each group, the program tries to interpret the 5 eigenphase sums as the "tail" of a resonance, and estimate the width and the position of the resonance centre. These estimates are then used to choose the next group of 5 energies. This option can be useful if a reasonably good estimate of the resonance energy is already available, and in favourable cases may succeed in converging towards a narrow resonance from as far as 10^5 widths away. However, convergence is *not* guaranteed, and it is not usually useful to do more than 3 sets of 5 energies in a single run. Furthermore, once the eigenphase sums span the resonance (or come very close to it), the automated algorithm fails to provide further improvement. At this point the resonance should be characterised using IECONV $= 4$ or 5, or in difficult cases by performing additional calculations on an appropriate equally spaced grid of energies.

Searches for resonances in the eigenphase sum as a function of energy should usually be done with log-derivative propagators, since they have good stability in the presence of closed channels. Since many energies are needed to characterise a resonance, it is usually most efficient to use the ISCRU option to save energy-independent matrices on a scratch file. For calculations with JTOTU $=$ JTOTL and IBFIX > 0, the ISCRU file from one run may be used as input for the next run at a different set of energies, as described in section 8.11.

9.10 Low-energy collision properties

9.10.1 Scattering lengths/volumes

MOLSCAT calculates energy-dependent complex scattering lengths (or volumes or hypervolumes for $L > 0$) using Eq. 2.14 as described in section 2.6.1. These are output in units (RUNIT)^n, where $n = 1$ for lengths ($L = 0$ channels), $n = 3$ for volumes ($L = 1$ channels) and $n = 4$ for hypervolumes ($L = 2$ channels). They are calculated only for 'low-energy' channels

where the wavevector k is less than 0.01 (RUNIT)$^{-1}$. This rather arbitrary threshold value may be changed by altering the value of the parameter AWVMAX in subroutine DRIVER (in the file mol.driver.f).

9.10.2 Scanning the scattering length/volume over an EFV

If IFCONV = 0 (the default), MOLSCAT performs a scan from FLDMIN to FLDMAX in steps of DFIELD. This option may be used with any value of NNRG.

9.10.3 Characterising a resonance in the scattering length as a function of EFV

At low collision energy, scattering resonances appear as characteristic features in the scattering length as a function of external field. In the absence of inelastic scattering, the scattering length shows a simple pole, but in the presence of inelasticity the behaviour is more complicated [14].

If IFCONV = 1, 2 or 3, MOLSCAT attempts to converge on and characterise a resonance in the scattering length/volume, as a function of the varying EFV, for the incoming channel with open-channel index ICHAN (section 9.3). It calculates scattering lengths/volumes at 3 different values (FLDMIN, FLDMAX, and [FLDMIN + FLDMAX]/2), and uses the algorithms of Frye and Hutson [12, 13] to converge on and characterise the resonance.

This option requires NNRG = 1.

IFCONV specifies the type of resonance, and which algorithm is used to characterise it:

IFCONV = 1 indicates that the resonance is elastic, with a pole in scattering length,

$$a(B) = a_{\mathrm{bg}} \left(1 - \frac{\Delta}{B - B_{\mathrm{res}}} \right), \tag{9.4}$$

where B represents the varying EFV. In this case the program uses the elastic procedure of ref. [13]. The parameters obtained are the pole position B_{res}, the resonance width Δ and the real background scattering length a_{bg}.

IFCONV = 2 indicates that the resonance is weakly decayed, meaning that the pole is suppressed and there is a peak in inelastic scattering at resonance, but that there is no significant background inelasticity away from resonance. In this case the program uses the weakly inelastic procedure of ref. [13]. the parameters obtained are as for IFCONV = 1 but with the addition of the resonant scattering length a_{res}, which is real and may be processed to obtain an inelastic width Γ_{inel}.

IFCONV = 3 indicates that the resonance is strongly decayed, meaning that there is significant background inelasticity away from resonance. The resonant peak in inelastic scattering is then asymmetric. In this case the program uses the fully complex procedure of ref. [13]. The parameters obtained are B_{res}, Δ and the real and imaginary parts of a_{res} and a_{bg}.

The values IFCONV = 4 and 5 are also implemented, to converge on resonances in the eigenphase sum and in a single S-matrix element as a function of EFV, by analogy with IECONV = 4 and 5 described in section 9.9. It should be noted that the value of $B_{\rm res}$ obtained in this way differs from that obtained from the scattering length, except at zero kinetic energy.

For any positive value of IFCONV, the variables TLO, THI and XI set values for the parameters $t_{\rm lo}$, $t_{\rm hi}$ and ξ of ref. [12].† The considerations that apply to the choice of TLO, THI and XI are the same as described in section 9.9; the defaults are usually adequate, except for wide resonances. The characterisation terminates when one of the three points is within DTOL of the estimated value of $B_{\rm res}$ and the other two are within the ranges prescribed by TLO, THI and XI. It also ends if MXLOC (set in module sizes) propagations are performed without convergence. The current estimates of the resonance parameters are printed at each step.

By default the algorithms neglect variation in $a_{\rm bg}(B)$. If the B-dependence can be estimated independently, a user-supplied routine BCKGRD may be provided to subtract it from the calculated scattering length.

In earlier work, a resonance width Δ_0 was sometimes defined such that $a(B_{\rm res} + \Delta_0) = 0$, so that the scattering length crosses zero a distance Δ_0 from the pole. This is equivalent to the definition based on Eq. 9.4 when $a_{\rm bg}$ does not vary across the width of the resonance, as is approximately true for most narrow resonances. However, Δ_0 does not properly capture the behaviour near the pole when there is significant variation in $a_{\rm bg}(B)$. It may in principle be obtained from the present algorithm by setting XI to a small value and THI = +1.0 for a resonance with positive Δ or THI = −1.0 for a resonance with negative Δ. However, such a large value of THI may give poor convergence for a wide resonance. If this occurs, and the value Δ_0 based on the zero crossing is truly required, it may be obtained by locating the position of the pole with a smaller value of |THI| and converging separately on the position of the zero crossing as described in section 9.10.4.

9.10.4 Converging on a specific value of the scattering length/volume

If IFCONV = −1, MOLSCAT attempts to converge on a value of the EFV where the scattering length in open channel ICHAN satisfies $a -$ AZERO $= 0$. This can be used to converge on a zero crossing if AZERO = 0.0. Convergence is attempted only if the value of $a -$ AZERO changes sign between FLDMIN and FLDMAX. Convergence uses the Van Wijngaarden-Dekker-Brent method [16] and terminates when the predicted step is less than DTOL.

This option requires NNRG = 1.

9.10.5 Effective range

In the absence of inelastic scattering, the s-wave scattering length a_0 is real. The near-threshold dependence of the s-wave scattering phase shift η on kinetic energy $E_{\rm kin}$ or wavevec-

†The implementation from version 2020.0 onwards differs slightly from that in version 2019.0 and 2019.1, with TLO, THI and XI replacing TOLMIN and TOLMAX.

tor k is often characterised by an effective-range expansion at small collision momentum

$$k \cot \eta(k) = -\frac{1}{a_0(0)} + \tfrac{1}{2} r_{\text{eff}} k^2 + \cdots \tag{9.5}$$

Re-expressing this using the definition $a_0(k) = -\tan \eta(k)/k$ gives two different expressions for the scattering length in terms of the effective range r_{eff}:

$$[a_0(k)]^{-1} = [a_0(0)]^{-1} - \tfrac{1}{2} r_{\text{eff}} k^2 + \cdots , \tag{9.6}$$

or

$$a_0(k) = a_0(0) + \tfrac{1}{2} r_{\text{eff}} [a_0(0)]^2 k^2 + \cdots . \tag{9.7}$$

At external fields far from a resonance or zero crossing in a_0, either of these relationships may be used to evaluate r_{eff}, using finite differences between scattering lengths evaluated at different kinetic energies. However, Eq. 9.6 is numerically unstable near a zero crossing and Eq. 9.7 is numerically unstable near a pole. For this reason, for each energy after the first, MOLSCAT calculates the effective range for open channel ICHAN from both

- a quadratic expansion of $1/[a_0(k)]$ at ENERGY(i) ($i > 1$) and ENERGY(1) and
- a quadratic expansion of $a_0(k)$.

ENERGY(1) should be small enough for $a_0(k)$ to be very close to $a_0(0)$ but not so small that numerical noise dominates the evaluation of $1 - S_{00}$.

This option requires NNRG to be greater than 1 and so must be done separately from characterisation of resonances or convergence on specific values. The effective range is calculated for the second and subsequent energies, when the energy is low enough that the scattering length itself is calculated, as described in section 9.10.1.

9.11 Scattering wavefunctions

If IWAVE > 0 and IPROPS $= 6$ (IPROPL $= 0$ or 6), MOLSCAT calculates the energy-normalised multichannel scattering wavefunction that is incoming only in channel ICHAN. The wavefunction is written on unit IPSI.

This feature is still under development, and its capabilities and input specifications may change in future, so it is not documented in detail here.

Chapter 10

Controlling bound-state calculations

10.1 State numbers and the node count

The multichannel node count used by BOUND and FIELD was introduced by Johnson [15]. It is defined as a function of energy and is equal to the number of bound-state solutions of a set of coupled equations that lie below that energy. It is evaluated during propagation of the coupled equations.

In versions of BOUND and FIELD before version 2019.0, the algorithm used required an additional propagation from either R_{min} or R_{max} to R_{mid}. From version 2019.0, this additional propagation is no longer performed; instead the node count is evaluated by summing the node counts from the outward and inward propagations and adding the number of negative eigenvalues of the log-derivative matching matrix.

The node count algorithm is quite reliable, but has been known to generate additional nodes in the classically forbidden region. For this reason, convergence algorithms based on the node count very occasionally fail.

BOUND and FIELD attempt to label each eigenvalue with a state number related to the node count. However, the node count as defined above *changes* at an eigenvalue. The programs define the state number as the node count at an energy just above the state.

The procedure used to assign a state number is not completely reliable, because rounding errors may cause the node count to change slightly above or below the eigenvalue. For this reason, the programs print warnings if the node count is not as expected, but this does not necessarily indicate that the bound-state position is in error and the programs do not terminate when it happens.

10.2 Locating bound states with BOUND

BOUND begins by propagating the log-derivative matrix at EMIN and EMAX and using the node counts to ascertain how many bound states to locate. The program searches for bound states with node counts between NODMIN and NODMAX that lie between EMIN and EMAX. It proceeds

initially by bisection, until it has identified a range of energies within which the node count changes by exactly 1. It continues to use bisection until this range is smaller than 100×DTOL, or until the minimum eigenvalue of the matching matrix changes from negative at the low-energy bound to positive at the high-energy bound. At this point it switches to using the Van Wijngaarden-Dekker-Brent algorithm [16] to converge on a bound state. Convergence terminates when the predicted step size is less than DTOL (in &INPUT energy units). The number of energy values allowed in converging on each bound state is limited by the internal variable NITER, which is currently set to 20. This value should be sufficient if RMATCH is chosen appropriately, unless exceptionally stringent convergence is required.

In special circumstances it may be desirable to carry out a scan of the node count and the smallest eigenvalue of the matching matrix as a function of energy. This is done by setting the absolute value of DNRG to less than the absolute value of EMAX − EMIN. BOUND then scans over energies in the range EMIN to EMAX using a step size of DNRG, without attempting to converge on eigenstates.

The namelist item MXCALC limits the number of energy values in a complete run.

10.3 Locating bound states with FIELD

FIELD reverses the order of the loops over energy and EFV value, and locates values of the EFV at which bound states with a specified energy exist. FIELD is particularly useful for estimating resonance positions at the lowest threshold of a particular symmetry, for use in subsequent scattering calculations, and also for mapping bound states whose energies vary very fast with the EFV.

An additional complication for FIELD is that the node count is not guaranteed to be a monotonic function of the EFV (though it often is, at least locally). This arises because states may pass through the energy of the calculation from either higher or lower values of the EFV. If the absolute value of DFIELD is less than the absolute value of FLDMAX − FLDMIN, FIELD scans over EFV values in the range FLDMIN to FLDMAX using a step size of DFIELD, without attempting to converge on eigenstates. This enables the user to identify EFV ranges within which the node count increases or decreases monotonically.

The algorithm used to converge on bound states is essentially the same as described above for BOUND: FIELD calculates the node counts at FLDMAX and FLDMIN. It then searches for and attempts to converge on bound states with node counts between NODMIN and NODMAX that lie between FLDMIN and FLDMAX. Convergence for each bound state terminates when the predicted step size is less than DTOL (which is interpreted as having the same units as the EFV concerned).

10.4 Bound-state wavefunctions

If IWAVE ≠ 0 and the LDMD propagator (IPROPS = IPROPL = 6) is used, then once a bound state has been located, BOUND or FIELD repeats the propagation at the converged energy or EFV, saving the log-derivative matrices to a temporary file, and then propagates the

wavefunction calculated at $R_{\rm match}$ back out to $R_{\rm min}$ and $R_{\rm max}$, as described in ref. [17]. The alternative extended Simpson's rule is used to find the normalisation constant [65].

The wavefunction is written on unit IPSI. The format of the wavefunction file is not yet finalised, so is not documented here, but is largely self-explanatory.

10.5 Expectation values

In addition to calculating bound-state energies, BOUND implements the calculation of expectation values using the finite-difference approach [66]. After a bound state is located at energy $E^{(0)}$, BOUND repeats the calculation with a small perturbation $a\hat{A}(R)$ added to the Hamiltonian to obtain a modified energy $E(a)$. From perturbation theory,

$$E_n(a) = E_n^{(0)} + a\langle\hat{A}\rangle_n + \mathcal{O}(a^2), \tag{10.1}$$

where $\mathcal{O}(a^2)$ are second-order terms. The finite-difference approximation to the expectation value $\langle\hat{A}\rangle_n$ is

$$\langle\hat{A}\rangle_n = \frac{E_n(a) - E_n^{(0)}}{a}, \tag{10.2}$$

and is accurate to order a.

For built-in interaction types, BOUND can calculate expectation values of an operator $\hat{A}$ that is made up of a product of one of the angular functions in the potential expansion and a power of R; see the documentation on potential expansions for each interaction type (section 11.3). More complicated functions of R can be handled by modifying subroutine PERTRB. For coupling cases implemented in plug-in basis-set suites, any required operator can be implemented in the VL array.

The expectation values to be calculated are specified by a variable NPERT and arrays IPPERT, NPOW, DELTA and FACTOR in namelist &INPUT. For each bound state located, the program attempts to calculate NPERT (up to 20) expectation values; if the operator whose coupling matrix (in the VL array) is A(IPPERT(IP)), the IPth value calculated is the expectation value of

$$\texttt{FACTOR(IP)} \times \texttt{A(IPPERT(IP))}/R^{\texttt{NPOW(IP)}}$$

The program applies a perturbation of DELTA(IP) $\times$ A(IPPERT(IP)) in order to use the finite-difference approximation. Small values of DELTA give smaller higher-order contributions to the result, but poorer numerical stability; DELTA $= 0.001$ cm^{-1} usually gives good results for operators with matrix elements of order of unity.

If IPPERT(IP) indexes an operator for an EFV and NPOW(IP) $= 0$, DELTA(IP) is interpreted in units of the EFV, and the program calculates the derivative of the bound-state energy with respect to the EFV, for both the absolute energy and the energy relative to threshold.

If the EFV is handled by setting NDGVL > 0 in a plug-in basis set suite, IPPERT(IP) must be set negative and the program then calculates the derivative of the bound-state energy with respect to EFV(-IPPERT(IP)).

10.6 Automatic convergence testing

If NCONV > 0, BOUND performs NCONV extra calculations of each eigenvalue or expectation value, with different values of RMIN, RMAX or DR. This is useful in testing the convergence with respect to these parameters, or in estimating the error due to the use of a finite step size. Which one of these lengths is varied is governed by ICON as follows:

ICON $= 1$ doubles the initial step size each time
ICON $= 2$ decreases RMAX by DRCON each time
ICON $= 3$ increases RMIN by DRCON each time

Note that, if BOUND does not succeed in converging on an eigenvalue for one value of the parameter concerned, the parameter is NOT changed before the next calculation. Make sure that BOUND can find the eigenvalue before attempting to test convergence.

10.7 Richardson extrapolation to zero step size

For propagators that use equally spaced or power-law steps, the error in bound-state energies due to a finite step size is proportional to a power of the step size (in the limit of small steps). This power is 4 for the LDJ, LDMD, LDMA and LDMG(CS4) propagators and 5 for the LDMG(MA5) propagator. It is thus possible to obtain an improved estimate of the bound-state energy by performing calculations with two different step sizes and extrapolating to zero step size. BOUND does this automatically and gives the extrapolated results as well as the ones using the step sizes specified when DR is varied using ICON $= 1$ as described in section 10.6.

If the short-range and long-range propagators are different, the power used for Richardson extrapolation is that appropriate for the short-range propagator. It is thus important to ensure that

- *either* the short-range and long-range propagators have the same step-size convergence properties, and both are used with either equally spaced steps or step sizes proportional to a power of R;
- *or* the short-range propagator is used throughout the well region, so that any errors due to the long-range propagator are negligible.

Chapter 11

Complete list of input parameters

The &INPUT item LASTIN controls whether the current set of namelist blocks is the last set to be used in the current run. If set to 0, the DRIVER routine loops back to the start, resetting all namelist items to their default values, and start a new set of calculations by reading in the next set of namelist blocks in the input file.

11.1 Items in namelist &INPUT

The namelist items are here listed alphabetically, followed by their default values in square brackets, then a short description of what they are and finally the section in which a more complete description may be found. If more than one default value is listed, they refer to different default values in the programs MOLSCAT, BOUND and FIELD using the colour coding described immediately below. If a namelist item is coloured, it is used by only a subset of the programs. The key is as follows:

if a keyword or default is red, it is used only by MOLSCAT,

if blue then it is used only by BOUND,

if green then it is used only by FIELD,

if cyan then it is used only by BOUND and FIELD,

if brown it is used only by MOLSCAT and FIELD, and

if purple it is used only by MOLSCAT and BOUND.

ADIAMN **[T]:** Controls whether adiabatic basis or primitive basis is used for boundary conditions at R_{min} (section 8.10).

ADIAMX **[T]:** Controls whether adiabatic basis or primitive basis is used for boundary conditions at R_{max} (section 8.10).

ALPHA1 **[1.0]:** Start value of step size for VIVS propagator (section 8.8.3).

ALPHA2 **[1.5]:** End value of step size for VIVS propagator (section 8.8.3).

AZERO **[0.0]:** Value of the (real part of the) scattering length/volume to converge upon (section 9.10.4).

BCYCMN **[−1.0]:** Controls value used in construction of log-derivative matrix for channels that are closed at $R_{\rm min}$ (section 8.10).

BCYCMX **[−1.0]:** Controls value used in construction of log-derivative matrix for channels that are closed at $R_{\rm max}$ (section 8.10).

BCYOMN **[unset]:** Controls value used in construction of log-derivative matrix at channels that are open at $R_{\rm min}$ (section 8.10).

BCYOMX **[0.0]:** Controls value used in construction of log-derivative matrix at channels that are open at $R_{\rm max}$ (section 8.10).

DEGTOL **[10^{-10}]:** Criterion used to test degeneracy, given in &INPUT energy units; see sections 9.2 and 9.6.

DELTA **[0.001]:** Array of step sizes used in the calculation of expectation values by finite differences (section 10.5).

DFIELD **[1.0/10^{30}]:** Step size for varying EFV (sections 7.1, 7.2, 9.10.2 and 10.3).

DNRG **[0.0/10^{30}]:** Step size for energies (sections 6.2, 9.9 10.2).

DRAIRY**:** Deprecated synonym for DRL.

DRCON **[0.1]:** Size of change to RMID or RMAX used to test convergence in MOLSCAT (section 9.5) and BOUND (section 10.6).

DRL **[unset]:** Initial value of δR used for the long-range part of the propagation (if appropriate) (sections 8.6 and 8.6.5).

DRS **[unset]:** Initial value of δR used for the short-range part of the propagation (if appropriate) (sections 8.6 and 8.6.5).

DR**:** Deprecated synonym for DRS.

DRMAX **[5.0]:** Maximum allowed step size for VIVS propagator (section 8.8.3).

DRNOW**:** Deprecated synonym for DR.

DTOL **[0.3/10^{-7}]:** Convergence criterion used for convergence on bound-state energies in BOUND (sections 10.2), bound-state EFVs in FIELD (section 10.3), and the positions of EFV-dependent Feshbach resonances (section 9.10.3) and specific values of scattering lengths (section 9.10.4) in MOLSCAT. Also applied to the convergence of diagonal (elastic) cross sections with respect to $J_{\rm tot}$ (section 9.6.3).

ECTRCT **[-10^{30}]:** Energy cutoff for contraction of basis set in RMAT propagator (section 4.10).

EMAX **[10^{30}]:** Upper end of energy range for calculations in BOUND (sections 6.5, 10.2).

EMIN **[-10^{30}]:** Lower end of energy range for calculations in BOUND (sections 6.5, 10.2).

ENERGY **[0.0]:** Array of energies for which calculations are requested (sections 6.2, 9.7, 9.9 and 9.10.5).

EPL **[0.0]:** Estimated maximum depth of the interaction potential in the long-range region; used only when the initial step size of the long-range propagator is determined from STEPL (section 8.6). Note that EPL is in &INPUT energy units, *not* &POTL energy units.

EPS **[0.0]:** Estimated maximum depth of the interaction potential in the short-range region; used only when the initial step size of the short-range propagator is determined from STEPS (section 8.6). Note that EPS is in &INPUT energy units, *not* &POTL energy units.

EREF **[0.0]:** Reference energy used for input and output of energies (scattering energy, depth of bound states) (section 6.6).

EUNITS **[1]:** Integer to choose units of energy for quantities in namelist &INPUT from the list in section 6.1.

EUNIT **[1.0]:** Units of energy (in cm^{-1}) if EUNITS is set to 0.

FACTOR **[1.0]:** Array of factors that calculated expectation values are multiplied by (section 10.5).

FIELD **[0.0]:** Array of values of varying EFVs for which calculations are requested (section 7.1).

FIXFLD(MXEFV) **[0.0]:** Array of values of fixed EFVs (section 7.1).

FLDMAX **[0.0]:** Upper end of range of varying EFV (sections 7.1, 7.2, 9.10.2, 9.10.3, 9.10.4 10.3).

FLDMIN **[0.0]:** Lower end of range of varying EFV (sections 7.1, 7.2, 9.10.2, 9.10.3, 9.10.4, 10.3).

IALFP **[F]:** Controls whether ALPHA2 is used as end value for step size in VIVS propagator (section 8.8.3).

IALPHA **[6]:** Controls whether step size can be variable in VIVS propagator (section 8.8.3).

IBDSUM **[0]:** Unit number for summary of bound-state locations (sections 13.1 and 12.7.1).

IBFIX **[0]:** If positive, lower bound for symmetry block index. If greater than IBHI, restricts symmetry block index to this value alone (section 4.9).

IBHI **[0]:** Highest value of symmetry block index for which calculations are required (section 4.9).

ICHAN **[1]:** Index of open channel to use in characterising an EFV-dependent Feshbach resonance (section 9.10.3), converging on an EFV with a specified value of the scattering length (section 9.10.4), calculating the effective range (section 9.10.5) or calculating a scattering wavefunction (section 9.11).

ICON **[1]:** Controls which variable is to be varied in convergence tests in MOLSCAT (section 9.5) and BOUND (section 10.6).

ICONVU **[0]:** Unit number for storing or retrieving an S matrix for use in convergence tests in MOLSCAT (section 9.5).

IDIAG **[F]:** Master control for perturbation correction type in VIVS propagator (section 8.8.3).

IECONV **[0]:** Specifies a scan of energies or a method for converging on a scattering resonance as a function of energy (section 9.9).

IFCONV **[0]:** Specifies a scan of scattering length as a function of EFV (section 9.10.2), the procedure to be used for convergence on a low-energy Feshbach resonance (section 9.10.3), or convergence on an EFV with a specified value of the scattering length (section 9.10.4).

IFVARY(MXEFV) **[min{1, NEFV}]:** index (or indices) of varying EFVs (section 7.1).

IFEGEN **[0]:** Control of generation of total energies for calculation of line-shape cross sections (section 9.7).

IFLS**:** Deprecated synonym for NLPRBR.

ILDSVU **[0]:** Unit number for output of log-derivative matrix (section 13.7).

IMGSEL **[4]:** Selects the method used for the symplectic (LDMG) propagator (section 8.8.4).

INTFLG**:** Deprecated variable to control propagator selection.

IPARTU **[0]:** Unit number for partial cross sections (not currently implemented; see sections 9.6.1 and 13.4).

IPERT **[T]:** Master control for inclusion of perturbation corrections in VIVS propagator (section 8.8.3).

IPHSUM **[0]:** Unit number for summary of eigenphase sums and scattering lengths (section 13.2).

IPPERT **[1,2,3,...]:** Array of indices of operators for which expectation values are to be calculated (section 10.5).

IPRINT **[2]:** Controls extent of printed output; see chapter 12.

IPROPL **[9/6]:** Propagator code for propagation between R_{mid} and R_{max} (section 8.1).

IPROPS **[6]:** Propagator code for propagation between R_{min} and R_{mid} (section 8.1).

IPSI **[109]:** Unit number for output of wavefunction; see section 10.4 for bound-state wavefunctions and section 9.11 for scattering wavefunctions.

IPSISC **[108]:** Unit number for internal storage of the wavefunction before reordering (scratch file) (sections 10.4 and 9.11).

IREF **[0]:** Index of the threshold whose energy is used as the reference energy (section 6.6).

IRMSET **[9]:** If positive, is used in choice of starting point for propagations (section 8.5).

IRSTRT **[0]:** Indicates whether the calculation is a continuation of previous scattering calculations (section 9.6.2).

IRXSET **[0]:** Indicates whether propagation is to be extended beyond RMAX to the outermost turning point of the centrifugal potential (section 8.5).

ISAVEU **[0]:** Unit number for output of S or K matrices (sections 9.6.2, 9.9 and 13.5).

ISCRU **[0]:** Scratch unit number for energy-independent matrices (section 8.11).

ISHIFT **[F]:** Controls whether 2nd derivative is used to shift reference potential for VIVS propagator (section 8.8.3).

ISIGPR **[0]:** Controls printing of cross sections (section 12.6.11).

ISIGU **[0]:** Unit number for accumulated cross sections (section 13.3).

ISYM **[T]:** Controls whether R matrix is forced to be symmetric for VIVS propagator (section 8.8.3).

IV **[T]:** Controls calculation of perturbation corrections due to potential for VIVS propagator (section 8.8.3).

IVP **[F]:** Controls calculation of perturbation corrections due to 1st derivative of interaction potential for VIVS propagator (section 8.8.3).

IVPP **[F]:** Controls calculation of perturbation corrections due to 2nd derivative of interaction potential for VIVS propagator (section 8.8.3).

IWAVE **[0]:** Controls whether wavefunctions are calculated (sections 9.11 and 10.4).

IWAVSC **[110]:** Unit number used for internal storage of arrays involved in generating wavefunctions (scratch file) (sections 9.11 and 10.4).

JSTEP **[1]:** Step length for loop over JTOT (section 4.9).

JTOTL **[0]:** Lowest value of JTOT (section 4.9).

JTOTU **[0]:** Highest value of JTOT (section 4.9).

KSAVE**:** Deprecated synonym for IPHSUM.

KSAVE**:** Deprecated synonym for IBDSUM in BOUND and FIELD.

LABEL [' ']: LABEL is a title for the run, up to 80 characters.

LASTIN [1]: Controls whether programs process another complete input data set after the current one (starting with another &INPUT block), or terminate; see page 106.

LINE [0]: Array of quartets of indices for levels, specifying spectroscopic lines for calculations of line-shape cross sections (section 9.7).

LMAX [0]: Highest L value for which IOS cross sections are accumulated (section 4.11).

LOGNRG [F]: If .TRUE., indicates that energies are to be generated in a geometric series (section 6.2).

LTYPE [−1]: Tensor order of spectroscopic transition involved in calculations of line-shape cross sections (section 9.7).

MAGEL [1]: Deprecated. Indicated which of the EFVs was referred to by FLDMIN, FLDMAX and DFIELD. This has now been replaced by IFVARY (with a different specification).

MHI: Deprecated synonym for IBHI.

MMAX [0]: Highest M value for which IOS cross sections are accumulated (section 4.11).

MONQN [MONQN(1) = −99999]: Array of values (typically quantum labels) used to specify the reference energy (section 6.6).

MSET: Deprecated synonym for IBFIX.

MXCALC [1000]: Maximum number of propagations (energies or EFV sets) in a run (sections 10.2 and 10.3).

MXPHI [1]: Number of values of the azimuthal angle for surface scattering calculations (ITYPE = 8, section 4.5.7).

MXSIG [0]: If positive, only cross sections between the first MXSIG open channels are calculated (section 9.6).

NCAC [14]: Convergence criterion for accumulation of cross sections (section 9.6.3).

NCONV [0]: Number of extra sets of calculations performed in convergence testing by MOLSCAT (section 9.5) or BOUND (section 10.6).

NFIELD [1]: Number of sets of EFVs for which calculations are requested (section 7.1).

NFVARY [-1]: Number of varying EFVs (section 7.1).

NGAUSS [3]: Number of Gaussian quadrature points generated for use in thermal averaging of cross sections (section 6.3).

NGMP [(8,1,16)]: Specifies numbers of quadrature points used in WKB integration (section 8.8.6).

NLPRBR **[0]:** Number of line-shape cross sections to be calculated, each specified by a pair of spectroscopic lines (section 9.7).

NNRG **[1]:** Number of values in ENERGY array (section 6.2).

NNRGPG **[1]:** Size of group of energies to be considered together in testing convergence of cross sections with respect to JTOT (section 9.6.3).

NODMAX **[99999]:** Upper bound of node count for bound states to be located (sections 6.5, 10.2, 10.3).

NODMIN **[0]:** Lower bound of node count for bound states to be located (sections 6.5, 10.1, 10.2, 10.3).

NPERT **[0]:** Number of operators for which expectation values are to be calculated (section 10.5).

NPOW **[0]:** Array of powers of R to be used in calculating expectation values (section 10.5).

NSTAB **[5]:** Controls how often linear independence is re-established for DV propagator (section 8.8.1).

NTEMP **[0]:** Number of temperatures for which energies are to be generated for use in (external) thermal averaging (section 6.3).

NUMDER **[F]:** Controls whether derivatives of potential coefficients are calculated numerically when required; needed only for the VIVS propagator (section 8.8.3) and for calculating nonadiabatic matrix elements with the LDMA propagator (section 8.8.4).

OTOL **[0.005]:** Convergence criterion applied to convergence of off-diagonal (usually inelastic) cross sections with respect to JTOT (section 9.6.3).

PHILW **[0.0]:** First value for azimuthal angle in surface scattering calculations (ITYPE = 8, section 4.5.7).

PHIST **[0.0]:** Step size for azimuthal angle in surface scattering calculations (ITYPE = 8, section 4.5.7).

POWRL [1.333̇ **or 3.0**]**:** Power used in step-size algorithm for RMAT and AIRY propagators at long range (sections 8.6.2, 8.8.2 and 8.8.5).

POWRS **[0.0 or 3.0]:** Power used in step-size algorithm for RMAT and AIRY propagators at short range (sections 8.6.2, 8.8.2 and 8.8.5).

POWRX**:** Deprecated synonym for POWRL.

PRNTLV**:** Deprecated synonym for IPRINT.

RCTRCT **[0.0]:** If greater than 0.0, value of R at which basis set is contracted in RMAT propagator (section 4.10).

RMATCH **[unset]:** Value of R at which log-derivative matrices for outwards and inwards propagation parts are matched; see sections 8.5 and 8.6.5. If left unset, set internally to RMID (if set).

RMAX **[10.0]:** Maximum value of R for propagation (sections 8.5, 8.6.5, 8.8.2, 8.8.3 and 8.8.5).

RMID **[10^{30}/RMATCH]:** Value of R at which propagation method switches for propagations which comprise more than one part (sections 8.5, 8.6.5, 8.8.3 and 8.8.5). Also used as the value where the step size starts to increase in the RMAT propagator (section 8.8.2). If left unset in BOUND or FIELD, set internally to RMATCH (if set).

RMIN **[0.8]:** Minimum value of R for propagation (sections 8.5, 8.6.5).

RVFAC **[0.0]:** If set greater than 0.0, controls how R_{mid} is chosen.

RVIVAS**:** Deprecated synonym for RMID.

SCALAM **[1.0]:** Value of the interaction potential scaling factor used in the current calculation (section 7.2).

STEPL **[−10.0]:** If positive, number of steps per half wavelength for long-range propagator (section 8.6). Otherwise DRL is used to set the step length.

STEPS **[−10.0]:** If positive, number of steps per half wavelength for short-range propagator (section 8.6). Otherwise DRS is used to set the step length.

TEMP **[0.0]:** Array of temperatures for which energies are to be generated for use in (external) thermal averaging (section 6.3).

THETLW **[0.0]:** First value for polar angle in surface scattering calculations (ITYPE = 8, section 4.5.7).

THETST **[0.0]:** Step size for polar angle in surface scattering calculations (ITYPE = 8, section 4.5.7).

THI **[1.0]:** controls positioning of the outermost of the 3 target points used in characterising a scattering resonance as a function of either energy (section 9.9) or EFV (section 9.10.3).

TLO **[−0.1]:** controls positioning of the second nearest of the 3 target points used in characterising a scattering resonance as a function of either energy (section 9.9) or EFV (section 9.10.3).

TOLHIL **[0.0001]:** Controls step size for variable-step propagators when used at long range (sections 8.8.2, 8.8.3, 8.8.5).

TOLHIS **[0.0001]:** Controls step size for variable-step propagators when used at short range (sections 8.8.2, 8.8.3, 8.8.5). Also used as the convergence criterion for the WKB phase shift (section 8.8.6).

TOLHI**:** Deprecated synonym for TOLHIS.

URED **[no default]:** Reduced mass for calculation (chapter 4).

WKBMN [.TRUE.]: Flag to use WKB boundary conditions in closed channels at $R_{\rm min}$ (section 8.10).

WKBMX [.TRUE.]: Flag to use WKB boundary conditions in closed channels at $R_{\rm max}$ (section 8.10).

XI [0.25]: controls the tolerance for the positioning of the two outer target points used in characterising a scattering resonance as a function of either energy (section 9.9) or EFV (section 9.10.3).

XSQMAX [10000]: Controls application of perturbation correction for VIVS propagator (section 8.8.3).

11.2 Items in namelist &BASIS

The parameters input in namelist &BASIS specify the interaction type, the quantum numbers and energies of the levels to be used in the basis set, and the dynamical approximations (if any) to be used in constructing the coupled equations.

A [0.0]: Array of dimension 2. The rotational constant about the x axis for a symmetric or asymmetric top; see sections 4.5.6, 4.5.4 and 4.5.5. A(2) is solely for use in plug-in basis-set routines.

ALPHAE [0.0]: Array of dimension 2. The vibrational dependence of the rotational constant for a vibrotor; see sections 4.5.1, 4.5.2, 4.5.3 and 4.5.6.

B [0.0]: Array of dimension 2. The rotational constant about the y axis for a symmetric or asymmetric top; see sections 4.5.6, 4.5.4 and 4.5.5. B(2) is solely for use in plug-in basis-set routines. Note that B is *not* the correct namelist item for the rotational constant for a linear rotor or vibrotor.

BCT [F]: Logical variable used to indicate that the centrifugal potential for the BCT Hamiltonian is to be used (section 4.12).

BE [0.0]: Array of dimension 2. The rotational constant for a linear rotor or vibrotor. See sections 4.5.1, 4.5.2, 4.5.3 and 4.5.6.

C [0.0]: Array of dimension 2. The rotational constant about the z axis for a symmetric or asymmetric top; see sections 4.5.6, 4.5.4 and 4.5.5. C(2) is solely for use in plug-in basis-set routines.

DE [0.0]: Array of dimension 2. The centrifugal distortion constant for a linear vibrotor; see sections 4.5.1, 4.5.2, 4.5.3 and 4.5.6.

DJ [0.0]: The centrifugal distortion constant D_J for a symmetric or asymmetric top; see sections 4.5.4 and 4.5.5.

DJK [0.0]: The centrifugal distortion constant D_{JK}; see sections 4.5.4 and 4.5.5.

`DK` **[0.0]:** The centrifugal distortion constant D_K; see sections 4.5.4 and 4.5.5.

`DT` **[0.0]:** The spherical top tetrahedral centrifugal distortion constant d_t; see section 4.5.5.

`ELEVEL` **[0.0]:** Array of dimension `MXELVL` (which is set in module `sizes` to be 1000). The energy levels for basis functions specified in `JLEVEL`. See section 4.5

`EMAX` **[0.0]:** If `EMAX` > 0.0, it is used to limit the selection of pair levels for `ITYP` $= 4$, `ITYP` $= 6$ and `ITYP` $= 8$; see sections 4.5.6, 4.5.5 and 4.5.7.

`EMAXK` **[0.0]:** If `EMAXK` > 0.0, it is used to limit the selection of basis functions basis functions for `ITYP` $= 8$; see section 4.5.7.

`EUNITS` **[1]:** Integer to choose units of energy for quantities in namelist `&BASIS` from the list in section 6.1; see section 4.4.

`EUNIT` **[1.0]:** Units of energy (in cm^{-1}) if `EUNITS` is set to 0.

`IASYMU` **[0]:** Unit number for input and output of asymmetric top functions; see sections 4.5.6 and 4.5.5.

`IBOUND` **[0]:** If `IBOUND` is set non-zero, it indicates that the basis set and centrifugal energy in calculations for `ITYPE` $= 21$ to 27 should be for helicity decoupling instead of L-labelled coupled states; see section 4.8. Also used by plug-in basis-set suites; see section 17.5.

`IDENT` **[0]:** Indicates whether interaction partners are identical (0 =non-identical, 1 =identical); see section 4.5.3.

`IOSNGP` **[0]:** Array of dimension 3: numbers of quadrature points over angles in IOS calculations; see section 4.11.

`IPHIFL` **[0]:** Controls the type of quadrature over $\phi_1 - \phi_2$ or χ in IOS calculations; see section 4.11.

`ISYM` **[−1]:** Array of dimension `MXSYMS` (set to 10 in module `sizes`), used for control of symmetry types included for symmetric and asymmetric tops; see sections 4.5.5, 4.5.6 and 4.5.4.

`ISYM2` **[−1]:** Array of dimension `MXSYMS` (set to 10 in module `sizes`), available for future expansion in analogy with `ISYM`. `ISYM2`(1) is also used to limit $L_{\max}$ (section 4.7).

`ITYPE` **[no default]:** see chapter 4.

`IVLU` **[0]:** if non-zero, indicates that the coupling matrices are stored on unit `IVLU` to save memory.

`J1MAX` **[0]:** Maximum value of J used in creating lists of pair levels for first structured particle. See section 4.5.3. It is equivalenced to `JMAX` and so can be specified using this name also.

`J1MIN` **[0]:** Minimum value of J used in creating lists of pair levels for first structured particle. See section 4.5.3. It is equivalenced to `JMIN` and so can be specified using this name also.

J1STEP **[1]:** Step used in creating lists of pair levels for first structured particle. See section 4.5.3. It is equivalenced to JSTEP and so can be specified using this name also.

J2MAX **[0]:** Maximum value of J used in creating lists of pair levels for second structured particle. See sections 4.5.3 and 4.5.6. It is equivalenced to KSET and to KMAX, and so may be specified using either of these names also.

J2MIN **[0]:** Minimum value of J used in creating lists of pair levels for second structured particle. See sections 4.5.3 and 4.5.6.

J2STEP **[1]:** Step used in creating lists of pair levels for second structured particle. See sections 4.5.3 and 4.5.6.

JHALF **[1]:** See section 17.3. Use in namelist deprecated. Should be set (if required) in BAS9IN (section 17.5).

JLEVEL **[0]:** Array of dimension MXJLVL (which is set to 4000 in module **sizes**). Contains labels for the NLEVEL monomer levels used in constructing the basis set. See section 4.5.

JMAX **[0]:** Maximum value of J used in creating lists of pair levels for first structured particle. See sections 4.5.1, 4.5.2, 4.5.4, 4.5.5 and 4.5.7. It is equivalenced to J1MAX and so may be specified using this name also.

JMIN **[0]:** Minimum value of J used in creating lists of pair levels for first structured particle. See sections 4.5.1, 4.5.2, 4.5.4, 4.5.5 and 4.5.7. It is equivalenced to J1MIN and so may be specified using this name also.

JSTEP **[1]:** Step used in creating lists of pair levels for first structured particle. See sections 4.5.1, 4.5.2, 4.5.4 ,4.5.5 and 4.5.7. It is equivalenced to J1STEP and so may be specified using this name also.

JZCSFL **[0]:** See section 4.8

JZCSMX **[−1]:** See section 4.8

KMAX **[0]:** If set greater than or equal to 0, limits the values of k for pair levels for ITYP $= 5$ to be less than or equal to KMAX; see section 4.5.4. It is equivalenced to J2MAX and KSET and so may be specified using these names also.

KSET **[0]:** If set less than 0, limits the values of k for pair levels for ITYP $= 5$ to be equal to |KSET|; see section 4.5.4. It is equivalenced to J2MAX and KMAX and so may be specified using these names also.

NLEVEL **[0]:** Number of sets of pair level quantum numbers to be read in. See chapter 4.

ROTI **[0.0]:** Array of dimension MXROTS (which is set to 12 in module **sizes**), containing rotational and vibrational constants for interaction partners. Most elements are equivalenced to other variables (see below), and so may be specified using other, more specific, names:

ROTI(1)≡	A(1),	BE(1)	ROTI(2)≡	A(2),	BE(2)
ROTI(3)≡	B(1),	ALPHAE(1)	ROTI(4)≡	B(2),	ALPHAE(2)
ROTI(5)≡	C(1),	DE(1)	ROTI(6)≡	C(2),	DE(2)
ROTI(7)≡	DJ,	WE(1)	ROTI(8)≡	DJK,	WE(2)
ROTI(9)≡	DK,	WEXE(1)	ROTI(10)≡	DT,	WEXE(2)

SPNUC **[0]:** If non-zero, used to calculate the statistical weights for interactions involving identical partners; see section 4.5.3.

WE **[0.0]:** Array of dimension 2. The equilibrium vibrational frequency of vibrating linear rotors; see sections 4.5.2 and 4.5.6.

WEXE **[0.0]:** Array of dimension 2. The anharmonicity constant of vibrating linear rotors; see sections 4.5.2 and 4.5.6.

WT **[0.0]:** Array of dimension 2 used to supply nuclear spin statistics weighting for identical partners; see section 4.5.4.

11.3 Items in namelist &POTL

The general-purpose version of subroutine POTENL supplied in this distribution reads a namelist block named &POTL. The parameters that may be input in &POTL are as follows; see chapter 5 for a full description:

CFLAG **[0]:** If 1, indicates that the interaction potential is expanded in modified spherical harmonics $C_{\lambda\kappa}$ rather than spherical harmonics Y_λ^κ; see section 5.1.4.

EPSIL **[1.0]:** Specifies the energy units of quantities input in &POTL and of the potential coefficients returned by POTENL; see section 5.2. When a routine VINIT/VSTAR or VRTP is supplied, EPSIL is usually coded there instead of being supplied in namelist &POTL. EPSIL must be specified in cm^{-1}. The value given to EPSIL does not affect the interpretation of energies input in &INPUT and &BASIS or output by the programs.

ICNSYM **[1]:** For nonlinear molecules, denotes symmetry about z-axis; see sections 5.1.4 and 5.3.2 (but note the caveat below in the description of IHOMO2). For example, for NH_3, ICNSYM = 3 indicates three-fold symmetry. For interaction potentials that are not expanded in angular functions this value should be set here or internally in VRTP; for interaction potentials that are expanded in angular functions, it is determined from the LAMBDA array.

IHOMO **[1]:** If 2, indicates homonuclear symmetry (reflection about $\theta = \pi/2$); see sections 5.1.1, 5.1.2 and 5.1.3. The default value of 1 indicates heteronuclear symmetry. For interaction potentials that are not expanded in angular functions, this value should be set here or internally in VRTP; see section 5.3.2). For interaction potentials that are expanded in angular functions, it is determined from the LAMBDA array.

ICNSY2 **[1]:** As IHOMO and ICNSYM, but for molecule 2. For historical reasons, ICNSYM may
IHOMO2 **[1]:** also be used to describe homonuclear symmetry of the second rotor for rigid rotor + rigid rotor interactions (ITYP = 3).

IVMIN [−1]: For ITYP = 2, if LMAX is used with LVRTP = .TRUE. to generate the LAMBDA
IVMAX [−1]: array internally, IVMIN must be given a nonnegative value to indicate the lowest vibrational level in the basis set. IVMAX, if greater than IVMIN, indicates the highest vibrational level; otherwise IVMAX is set equal to IVMIN; see section 5.1.2.

LAMBDA [0]: An array of labels for the MXLAM different terms included in the interaction potential; see section 5.1.

LMAX [−1]: When given non-negative values, the appropriate subset of these quantities
MMAX [−1]: specify the highest term(s) to include in the potential expansion. See sections
L1MAX: 5.1.1, 5.1.2, 5.1.4 and 5.1.3 for more details. IHOMO and ICNSYM (described
L2MAX [−1]: above) may be used to exclude values not allowed by symmetry.

LVRTP [.FALSE.]: If LVRTP = .FALSE., the interaction potential is specified in terms of its expansion in angular functions; see section 5.3.1. If LVRTP = .TRUE., subroutine VRTP is called as described in section 5.3.2 to evaluate the interaction potential at specified interaction coordinates. Note that LVRTP is forced to be .TRUE. if MXLAM ≤ 0.

MXLAM [0]: If positive, the number of terms in the expansion of the interaction potential.

MXSYM: Deprecated synonym for MXLAM.

NPTS [0]: Array specifying the numbers of Gauss integration points used in projecting the angular components of the interaction potential; see section 5.3.2.

NPT: Deprecated synonym for NPTS(1).

NPS: Deprecated synonym for NPTS(2).

NTERM [−1]: An array of MXLAM integers, describing how to evaluate the interaction potential if LVRTP = .FALSE.; see section 5.3.1. Each element of the NTERM array corresponds to one element in the expansion of the interaction potential.

If NTERM(i) < 0, this element of the potential array is evaluated using the VINIT/VSTAR mechanism.

If NTERM(i) is positive, this element of the potential array is evaluated as a sum of NTERM(i) exponential or inverse-power terms, specified by A, E and NPOWER.

NPOWER [0]: Array of integers. If zero, the corresponding term has the form A $\exp(-\mathtt{E} \times R)$, and if positive it has the form $\mathtt{A}/R^{\mathtt{NPOWER}}$.

A [0.0]: Array of prefactors for interaction potential terms.

E [0.0]: Array of inverse lengths that specify the range parameter in exponential potential terms.

RM [1.0]: specifies the units of length that are used by the potential routine, and throughout the programs if RUNIT is not specified in namelist &INPUT. RM is specified in units of Å. When a routine VINIT/VSTAR or VRTP is supplied, RM is often coded there instead of being supplied in namelist &POTL. Subsequent calls to POTENL handle distances in these units.

Chapter 12

Controlling the print level

The level of output written to the standard output channel (Fortran unit 6) is controlled by the integer variable IPRINT; sensible values of IPRINT vary from 1 (when only integral cross sections are required from MOLSCAT, or only details of converged energies or fields are required for bound-state calculations) to 40 (when debugging).

For MOLSCAT, state-to-state integral cross sections are printed if ISIGPR > 0 (regardless of the value of IPRINT), scattering lengths/volumes are printed if IPRINT ≥ 6, and complete S matrices are printed if IPRINT ≥ 11. Voluminous debugging output starts appearing at IPRINT $= 15$.

For bound-state calculations (BOUND and FIELD), located bound states are printed if IPRINT $=$ 1 or greater, IPRINT $= 5$ prints the larger components of the bound-state wavefunction at the matching point and IPRINT $= 6$ prints brief information on the progress of locating bound states. Voluminous debugging output starts appearing at IPRINT $= 12$.

A certain amount is always printed, regardless of the value of IPRINT, including:

- Any error conditions that cause the program to stop prematurely
- Warnings about namelist values that have had to be changed because of incompatibilities between them
- Some, but not all, other warnings
- Some output from older sections of infrequently used code, such as for IOS calculations and line-shape cross sections

Note that this is not a complete list.

12.1 Headers and loops

If IPRINT ≥ 1, all the programs print:

- a header;
- the run label;
- information about propagator choice, ranges and step size control;
- information about the generation of basis sets;

- information about energies and sets of EFVs;
- information about the interaction potential.

The programs then loop over JTOT, symmetry block IBLOCK, sets of EFVs and energies as required.

For each JTOT and symmetry block IBLOCK, the programs print

- details of the specific basis set used for that JTOT and IBLOCK (which for non-diagonal Hamiltonians is the primitive set) if IPRINT ≥ 5.

For each set of EFVs, the programs print

- for basis sets non-diagonal in $H_{\rm intl}$ or $\hat{L}^2$, information on the asymptotic basis set and how it relates to the primitive basis set, and the resulting thresholds if IPRINT ≥ 6 (10 for FIELD).

In MOLSCAT and BOUND, the loop over energies is inside the loop over field. At each energy, the programs print

- the energy relative to the reference energy EREF if IPRINT ≥ 7;
- the absolute energy if IPRINT ≥ 8 and EREF $\neq 0.0$.

In FIELD, the loop over EFVs is inside the loop over energy. At each EFV, FIELD prints

- the reference energy EREF if IPRINT ≥ 10;
- the absolute energy if IPRINT ≥ 9 and EREF $\neq 0.0$.

The behaviour of the programs differs within the innermost loop, and so does the output, but the description of individual propagators is common to all of them.

12.2 Basis sets and quantum numbers

The list of pair state quantum numbers stored in the array JSTATE, described in sections 4.2.1 and 4.2.2, is printed if IPRINT ≥ 1. If $H_{\rm intl}$ and $\hat{L}^2$ are diagonal, the pair level index and energy that correspond to each set of quantum numbers are also printed. Once the basis set has been chosen for a particular set of coupled equations (i.e., for a particular JTOT and symmetry block), the basis functions included are printed if IPRINT ≥ 5. For basis sets diagonal in $H_{\rm intl}$ and $\hat{L}^2$, each basis function corresponds to a scattering channel, and its energy is also given.

12.3 Coupling matrices

If IPRINT ≥ 26, the coupling matrices $\boldsymbol{\mathcal{V}}^{\Lambda}$ for the current JTOT and symmetry block are printed. The programs print a warning if all the elements of a particular coupling matrix are zero and IPRINT ≥ 14. Once all the coupling matrices have been calculated, the programs print the number of them that are all zero if IPRINT ≥ 10.

12.4 Reference energy, threshold energies and channel indices

If energies are specified relative to a non-zero reference energy as described in section 6.6, then the method of choosing the reference energy is printed if IPRINT ≥ 1. If the value of the reference energy is independent of the symmetry block and of the values of EFVS, it is also given. If it depends on the symmetry block or the values of EFVS, it is printed each time it is calculated if IPRINT ≥ 3 (for BOUND), 4 (for MOLSCAT) or 6 (for FIELD).

A list of the channel (or threshold) energies is printed if IPRINT ≥ 6 (for MOLSCAT and BOUND) or 10 (for FIELD).

A list of the open channels is printed if IPRINT ≥ 10.

12.4.1 Threshold energies for asymptotically non-diagonal basis sets

For asymptotically non-diagonal basis sets, the internal Hamiltonian is constructed and diagonalised before propagation in order to obtain the threshold energies. If $\hat{L}^2$ is diagonal, the basis set is first separated into sets corresponding to different values of L (or, if IBOUND $= 1$, values of CENT that are the same to within DEGTOL). The constant coupling coefficients, h_Ω and (later) the eigenvalues of the internal Hamiltonian are printed if IPRINT ≥ 10. For each set, the matrix of the internal Hamiltonian is printed if IPRINT ≥ 25. The eigenvalues are printed if IPRINT ≥ 15. The eigenvectors are printed if IPRINT ≥ 25.

If $\hat{L}^2$ is non-diagonal, it is treated as an extra operator as described in the following section.

12.4.2 Resolving degeneracies amongst threshold energies

If extra operators are used to resolve (near-)degeneracies between threshold energies, MOLSCAT works through $H_{\rm intl}$ and the extra operators in turn. The constant coupling coefficients h_Ω and the coefficients for the extra operators are printed if IPRINT ≥ 6. Any sets of eigenvalues of the current operator that are degenerate to within DEGTOL are printed if IPRINT ≥ 10. The print levels for matrices, eigenvalues and eigenvectors of the submatrices are as in section 12.4.1. Transformed submatrices are printed if IPRINT ≥ 30.

If the eigenvectors need to be reordered to match their ordering in the internal Hamiltonian, the complete set of reordered eigenvalues is printed if IPRINT ≥ 15 and the reordered eigenvectors are printed if IPRINT ≥ 25.

Once MOLSCAT has worked through all the operators, the eigenvalues for all operators are printed for each channel if IPRINT ≥ 6. The channel numbering in this list is the one used for the channel threshold energies described in section 4.2.2. A warning is printed if any channels are degenerate in all operators.

12.5 Propagations

If $R_{\rm min}$ and/or $R_{\rm max}$ have been altered from the input values as described in section 8.5, a message is printed to this effect if IPRINT ≥ 9. More detailed information about the search for a suitable value of $R_{\rm min}$ is printed if IPRINT ≥ 13 and IRMSET > 0.

All propagators (except the WKB integrator) print the values of R at which the propagation starts and ends, and the number of steps taken, if IPRINT ≥ 8. In addition:

- The WKB integrator prints the WKB phase shift and how many quadrature points were used to obtain it if IPRINT ≥ 4. It also gives progress information about the searches for turning points if IPRINT ≥ 13; see section 8.8.6.
- The DV propagator prints prints the values of R at which stabilisation is done if IPRINT ≥ 13; see section 8.8.1.
- The RMAT propagator prints the largest and smallest eigenvalues of the matching matrix at each step if IPRINT ≥ 20; see section 8.8.2.
- The VIVS propagator prints the value of R and an estimate of the size of the derivative of the irregular solution $f_2^{i'}(R)$ every time a new interval is started if IPRINT ≥ 13, and all the control data if IPRINT ≥ 20; see section 8.8.3.
- The LDMA propagator prints (at each step of the propagation) the lowest 30 adiabats if IPRINT ≥ 19 and the first 30 diagonal elements of $\hbar^2\mathbf{W}/2\mu$ if IPRINT ≥ 20, the first 9 radial potential coefficients $v_\Lambda(R)/$EPSIL if IPRINT ≥ 21, the interaction matrix $\mathbf{W}(R)$ (in reduced units) if IPRINT ≥ 22, the matrix of the nonadiabatic couplings d/dR between the adiabatic states if IPRINT ≥ 23 and the eigenvectors that define the adiabatic states if IPRINT ≥ 24.
- The AIRY propagator prints the maximum values of CDIAG and COFF and the number of steps in which they exceeded 5 times the accuracy tolerance TOLHI if IPRINT ≥ 12. It prints the individual steps that exceeded this criterion and the sizes of the smallest and largest steps if IPRINT ≥ 13. It prints the midpoints, sizes, and diagonal and off-diagonal correction terms for each step if IPRINT ≥ 20; see section 8.8.5.

All the values of IPRINT in this subsection are increased by 10 for IOS calculations.

12.6 Scattering calculations

12.6.1 S-matrix elements

The open-channel basis functions and wavevectors are printed if IPRINT ≥ 10, and S-matrix elements with square modulus greater than 10^{-20} are printed if IPRINT ≥ 11.

For IOS calculations, the S-matrix elements are printed if IPRINT ≥ 15.

12.6.2 Eigenphase sums

If IPHSUM > 0 and IPRINT ≥ 6, MOLSCAT prints the S-matrix eigenphase sum.

12.6.3 Scattering lengths/volumes

If IPRINT ≥ 6, MOLSCAT prints the scattering length/volume for each channel that has a low kinetic energy. For comparison, various approximate forms are additionally printed if IPRINT ≥ 26. These are (where n is 1 if $L = 0$, 3 if $L = 1$ and 4 otherwise):

$$a_L \approx -\frac{\arg S}{2k^n}; \qquad \text{and} \qquad a_L \approx \frac{(1-S)}{\mathrm{i}k^n}. \tag{12.1}$$

12.6.4 Convergence of S-matrix elements

If automatic convergence testing of S-matrix elements is requested, as described in section 9.5, convergence information is printed without additional print controls.

12.6.5 Cross sections (non-IOS calculations)

If ISIGPR > 0, some cross section information is printed in the main output if relevant: if IPRINT ≥ 3, partial cross sections are printed after every propagation. If IPRINT ≥ 11 the state-to-state integral cross sections accumulated thus far are also printed.

If ISIGU > 0 the state-to-state integral cross sections accumulated thus far are updated on unit ISIGU.

12.6.6 Cross sections (IOS only)

IOS total cross sections are printed regardless of the value of IPRINT, as is their average over orientations, which is equal to the total scattering $Q^t(0,0,0) - Q^s(0,0,0)$. IOS total elastic and inelastic scattering are also printed regardless of the value of IPRINT, as are the contributing dynamical factors $Q^{s \text{ or } t}(L, M_a, M_b)$.

The collision dynamics factors are also written to unit ISAVEU for every collision energy and value of JTOT if ISAVEU > 0.

Setting IPRINT to a positive value gives the following:

IPRINT ≥ 2 prints the collision energy and current value of JTOT.

IPRINT ≥ 10 prints the current contributions to the collision dynamics factors, together with the totals accumulated thus far.

IPRINT ≥ 13 prints the current contributions to the total cross sections, together with the totals accumulated thus far.

IPRINT ≥ 20 prints the current contributions to the T matrix with the totals accumulated thus far.

12.6.7 Line-shape cross sections

When calculating line-shape cross sections (section 9.7) without using the IOS approximation, the total accumulated line-shape cross sections are always printed at the end of the calculation. If IPRINT ≥ 1, the running totals are printed after each JTOT and IBLOCK. If IPRINT ≥ 4, the contributions from individual S-matrices are printed as they are calculated.

For calculations using the IOS approximation, no additional information is printed depending on the value of IPRINT.

12.6.8 Characterisation of a resonance or quasibound state as a function of energy or EFV

For characterisation of a resonance with IECONV $= 4$ or 5 (section 9.9) or IFCONV $= 1$ to 5 (section 9.10.3):

If IPRINT ≥ 2, the converged resonance location is printed.
If IPRINT ≥ 3, the resonance parameters at the final step are printed.
If IPRINT ≥ 5 and IECONV or IFCONV is 4 or 5, the partial widths are printed.
If IPRINT ≥ 6, the current estimates of the resonance parameters are printed after each step.
If IPRINT ≥ 7, information is printed about the logic used to choose the next energy or EFV, with slightly more detail if IPRINT ≥ 8.

For stepping towards a resonance with IECONV $= -5$ or NNRG < 0:

If IPRINT ≥ 6, the (groups of 5) energies and eigenphase sums used to estimate the position of an energy-dependent resonance, as described in section 9.9, are printed, together with the resulting estimate of the resonance position and width.
If IPRINT ≥ 10, all (3) current estimates of the resonance location and width are printed.
If IPHSUM > 0, the eigenphase curvatures and estimated location and width of an energy-dependent resonance are included on unit IPHSUM.

12.6.9 Locating the value of an EFV at which the scattering length/volume has a specific value

If IPRINT ≥ 4, information is printed about progress in converging on the EFV; see section 9.10.4.

12.6.10 Effective range

If IPRINT ≥ 1, MOLSCAT prints the effective range r_{eff}, as described in section 9.10.5.

12.6.11 State-to-state cross sections in main output file

The printing of total and partial cross sections to the main output file is controlled by the parameter ISIGPR. This must be set to 1 if printing of cross sections is required (2 to include coupled-states cross sections that are incomplete due to missing values of K because of JZCSMX).

12.7 Bound-state calculations

12.7.1 Bound-state positions in energy or EFV

If IPRINT ≥ 6, the energy (for BOUND) or the EFV (for FIELD) is printed for each propagation. If IPRINT ≥ 7, the method used to choose a new value for the next propagation and the resulting new value are printed, together with the resulting total node count and the eigenvalue of the matching matrix with the smallest absolute value.

If IPRINT ≥ 8, BOUND and FIELD print the node count for each propagation segment and the number of negative eigenvalues of the matching matrix (section 10.1). They print the eigenvalues themselves if IPRINT ≥ 9. They print the full matching matrix if IPRINT ≥ 12, and the log-derivative matrix at the end of each propagation part if IPRINT ≥ 15.

If IPRINT ≥ 1, BOUND and FIELD print the location of a bound state when convergence on it has succeeded. They print its absolute energy if IPRINT ≥ 8. They also print the larger components of the wavefunction at the matching point if $5 \leq$ IPRINT < 11 or *all* the components if IPRINT ≥ 11. If IPRINT ≥ 8 they print the CPU time taken to converge on that bound state.

If IBDSUM > 0, BOUND and FIELD print the state number and location (energy, EFV set) of located bound states to unit IBDSUM.

If performing a scan, BOUND and FIELD print the node count and smallest eigenvalue of the matching matrix at each energy (for BOUND) or EFV set (for FIELD) if IPRINT ≥ 1.

12.7.2 Expectation values

Expectation values (section 10.5) are printed if IPRINT ≥ 1.

12.7.3 Convergence testing

Results of any convergence testing are printed if IPRINT ≥ 1; see section 10.6.

12.7.4 Calculation of wavefunction

The wavefunction (section 10.4) is output on unit IPSI; see section 13.6.

BOUND and FIELD print the total normalisation integral if `IPRINT` ≥ 6, and the contribution from each basis function if `IPRINT` ≥ 8. The entire wavefunction is printed in the main output file if `IPRINT` ≥ 30.

Chapter 13

Auxiliary output and scratch files

The programs may produce a number of auxiliary files, depending on the values of parameters in namelist `&INPUT` (except `IASYMU`, which is controlled by namelist `&BASIS`). They also use scratch files under certain circumstances. The following is a complete list of these auxiliary files and the input parameters that control whether they are used. Direct-access files are indicated by DA. The programs that make use of each file are indicated by their initial letter(s). Each file is used only if the corresponding unit number is set to (or defaults to) a non-zero value.

I/O unit	Used by	unformatted	Use	See section
`IASYMU`	MBF	no	Asymmetric top rotor functions for `ITYP` = 4 or 6	4.5.6, 4.5.5
`IBDSUM`	BF	no	Summary of energies and EFV values for bound states	13.1
`IPHSUM`	M	no	Eigenphase sums and scattering lengths for low-energy scattering channel `ICHAN`	13.2
`ISAVEU`	M	yes	S matrices if `IPHSUM` = 0	13.5
			K matrices if `IPHSUM` > 0	13.5
`IPSI`	MBF	no	Wavefunctions	10.4, 9.11
`ILDSVU`	M	yes	Log-derivative matrix	13.7
`IVLU`	MBF	yes	Coupling matrices	13.8
`ICONVU`	M	yes	S matrices for use in convergence runs	9.5
scratch unit				
`ISIGU`	M	no, DA	Cross sections, updated after each S matrix is calculated	13.3
`IPSISC`	MBF	yes, DA	Scratch file for wavefunctions	10.4, 9.11
`IWAVSC`	MBF	yes, DA	Scratch file for wavefunctions	10.4, 9.11
`ISCRU`	MBF	yes	Propagator scratch unit	8.11

The following sections describe these files in more detail, except for those covered elsewhere.

13.1 Summary of bound states

BOUND and FIELD write a concise summary of converged bound-state energies or EFVs on IBDSUM. The summary includes the state number, together with a warning if it does not agree with the expected value (but see section 10.1).

13.2 Summary of eigenphase sums and low energy scattering lengths

Eigenphase sums are written to unit IPHSUM if IPHSUM > 0. Scattering lengths for channel ICHAN are written to IPHSUM if both ICHAN and IPHSUM are set greater than 0.

13.3 State-to-state integral cross sections

If ISIGU > 0, MOLSCAT maintains a (direct access) file containing the state-to-state integral cross sections accumulated thus far on unit ISIGU. This file is updated every time an S matrix is processed to give contributions to the cross sections, so it contains valid information about the run so far even if the program terminates abnormally.

Code to read the contents of this file is included in this release (subroutine RDSIGU), but is not executed.

13.4 Partial cross sections

The option to print partial cross sections to a separate file, on unit number IPARTU, is not implemented in MOLSCAT version 2020.0, but could be resuscitated if necessary.

13.5 S and K matrices

In addition to its main printed output on unit 6, MOLSCAT can also produce files containing S matrices and/or K matrices for subsequent processing by other programs. The S matrix output is compatible with programs DCS [2] (for differential cross sections) and SBE [3] (for generalised cross sections for transport and relaxation properties and Senftleben-Beenakker effects). The K matrix output can be read by program SAVER, which accumulates output from different runs and outputs it in a format suitable for program RESFIT [4] (which fits eigenphase sums to obtain the positions and widths of Feshbach resonances diagonal S-matrix elements to obtain partial widths).

If ISAVEU > 0, MOLSCAT saves *either* S matrices *or* K matrices on unit ISAVEU. If IPHSUM ≤ 0 it saves S matrices and if IPHSUM > 0 it saves K matrices.

The format of these files is described below. Some aspects of the formats have changed

between versions, and the files include an ISAVEU format version number IPROGM as described below. If desired the subroutines SKREAD (with entry points HDREAD, SLPRD and KLPRD) and SREAD can be used by other post-processor programs to read the headers and loop output, and take account of values of IPROGM from recent versions of MOLSCAT.

13.5.1 S matrices

MOLSCAT saves S matrices in an unformatted (binary) file. The results are written as single (logical) records (i.e., single unformatted WRITE statements), except for (8), which is described more fully below. Beginning with IPROGM = 14 (August 1994), NOPEN is in the record before the one in which it is used. Beginning with IPROGM = 17, values for EFVs are included in the record that starts with JTOT, and there are two additional records immediately before this one. The first of these contains integer variables relating to whether the basis set is diagonal or not, and the second gives the number of EFVs and their names and units.

If S matrices are output to the ISAVEU file, its contents are as follows:

1. LABEL, ITYPE, NSTATE, NQN, URED, IPROGM

   LABEL is the title of the run and is a character variable of length 80.

   ITYPE specifies the interaction type.

   NSTATE is the number of pair states in the basis set.

   NQN is the number of (quantum) labels per pair state.

   URED is the reduced mass in atomic mass units (or in units of MUNIT atomic mass units if MUNIT is not 1).

   IPROGM is the version number for the format of the output written to unit ISAVEU. IPROGM is distinct from the program version number and is 19 for MOLSCAT version 2020.0.

2. ((JSTATE(I,J), I = 1, NSTATE), J = 1, NQN)

   JSTATE(ISTATE,J) are the quantum numbers of state ISTATE. The meaning depends on ITYPE; see chapter 4.

3. NLEVEL, (ELEVEL(I), I = 1, NLEVEL)
   Number of pair levels and the contents of the ELEVEL array, which contains pair level energies indexed by INDLEV.

4. NDGVL, NCONST, NRSQ, IBOUND, ITPSUB

   NDGVL is the number of diagonal terms in the internal Hamiltonian that depend on EFVs.

   NCONST is the number of terms that contribute to a non-diagonal internal Hamiltonian.

   NRSQ is the number of terms that contribute to a non-diagonal $\mathcal{L}$ operator.

   IBOUND is 0 if the matrix elements for the operator $\mathcal{L}$ are just the diagonal values $L(L+1)$, and 1 if they are in the array CENT.

ITPSUB is an integer flag that may be set when ITYPE = 9 to specify the particular plug-in basis-set suite that produced the results.

5. NEFV, ISVEFV, (EFVNAM(IEFV), EFVUNT(IEFV), IEFV = 1, NEFV)

   NEFV is the number of EFVs (excluding potential scaling).

   ISVEFV is the index of the single varying EFV (set to NEFV + 1 for a proxy EFV).

   EFVNAM are the names of the EFVs (excluding potential scaling).

   EFVUNT are the units of the EFVs (excluding potential scaling).

6. NFIELD, NNRG, (ENERGY(I), I = 1, NNRG)
   Number of different sets of external fields to be looped over followed by the number and values of the scattering energies (cm^{-1}).

7. JTOT, INRG, IBLOCK, IFIELD, EN, (EFV(IEFV), IEFV = 0, NEFVP), EREF, &
   IEXCH, WT, NOPEN
   These describe a single scattering calculation:

   JTOT is the total angular momentum.

   INRG is the index of the energy in the list in 5 above.

   IBLOCK is the index of the symmetry block.

   IFIELD is the index of the current set of EFVs.

   EN is the current scattering energy (in cm^{-1}) relative to the reference energy EREF; it should equal ENERGY(INRG).

   EFV is the array of the current values of the EFVs. NEFVP is equal to NEFV unless EFVs are calculated from a proxy EFV, in which case it is NEFV + 1.

   EREF is the energy that EN is referenced to, so that EN + EREF is the total scattering energy.

   IEXCH is the exchange parity for identical molecules

   IEXCH = 0 no exchange symmetry

   IEXCH = 1 odd exchange symmetry

   IEXCH = 2 even exchange symmetry

   WT (if nonzero) is the statistical weight for the current values of JTOT and IBLOCK.

   NOPEN is the number of open channels in the S matrix.

8. (*index*(I), *L*(I), WV(I), I = 1, NOPEN)

   index(I) is a pointer to the arrays of quantum numbers and threshold energies.

   - For asymptotically diagonal basis sets, *index* is a pointer to the array JSTATE that contains pair state quantum numbers. JSTATE(*index*(I),NQN) is itself a pointer to the element of the array ELEVEL that contains the threshold energy for channel I.
   - For asymptotically non-diagonal basis sets, *index*(I) is the index of the element of the array ELEVEL that contains the threshold energy for channel I.

`L(I)` specifies the orbital angular momentum for channel `I`.

- If `IBOUND` = 0, it is the integer `L(I)`.
- If `IBOUND` = 1, it is the diagonal matrix element of $\hat{L}^2$ stored in `CENT(I)`. If `NRSQ` $\neq 0$, this value is obtained by diagonalising the matrix of $\hat{L}^2$.

`WV(I)` is the wavevector of channel `I` (Å^{-1}).

9. `SREAL`
10. `SIMAG`
    `SREAL(I)` and `SIMAG(I)` are the real and imaginary parts of the `NOPEN` by `NOPEN` S matrix. They are each written as a single record, listing only the lower triangle, i.e.,
    `((SREAL(I,J), J = 1, I), I = 1, NOPEN)`
    `((SIMAG(I,J), J = 1, I), I = 1, NOPEN)`

Records 7–10 are repeated for each S matrix calculated, looping over `IFIELD` (innermost), `INRG`, `IBLOCK` and `JTOT` (outermost):

```
DO JTOT = JTOTL, JTOTU, JSTEP
  DO IBLOCK = 1, NBLOCK
    DO INRG = 1, NNRG
      DO IFIELD = 1, NFIELD
        7.), 8.), 9.), 10)
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

`NBLOCK` depends on `ITYPE`. Note that not every S matrix necessarily exists. S matrices may be missing from the file either because there are no open channels for that energy, or because there was an error or convergence failure in the calculation.

Subroutines `SKREAD` (with entry points `HDREAD` and `SLPRD`) and `SREAD` are included in this distribution for use in reading the unformatted files that have been standard since version 11. Line 1 must be read by the program directly, following which a call to entry point `SHDRD` reads lines 2–6. Within loops over `JTOT`, `IBLOCK`, `INRG` and `IFIELD` the entry point `SLPRD` can be called to read the rest of the records. It uses `SREAD` to read the real and imaginary parts of the S matrices written in records 9 and 10.

13.5.2 K matrices

If K matrices are output to the `ISAVEU` file, the (unformatted) output is very similar to the output for S matrices described above. The first 8 records are exactly as described above for the S matrix file. They are followed by:

9. `(SREAL(I), SIMAG(I), I = 1, NOPEN*NOPEN, NOPEN+1)` are the diagonal elements of the S matrix.

10. ((AKMAT(I,J), J = 1, I), I = 1, NOPEN) is the K matrix. Only the lower triangle is written.

11. ESUM is the eigenphase sum.

Subroutines SKREAD (using entry points HDREAD and KLPRD) and SREAD are included in this distribution for use in reading the unformatted files that have been standard since version 11. Line 1 must be read by the program directly, following which a call to entry point HDREAD reads lines 2–6. Within loops over JTOT, IBLOCK, INRG and IFIELD the entry point KLPRD can be called to read records 7–10. It uses SREAD to read the K matrices written in record 10.

13.6 Wavefunctions (LDMD propagator only)

If IWAVE > 0, the multichannel wavefunction is saved on unit IPSI. For MOLSCAT this is the wavefunction for flux incoming only in scattering channel ICHAN. By default, IPSI $= 109$ and the file is formatted. This file is likely to be quite large as it contains the wavefunction at every point of the propagation. If an unformatted file is preferred, it may be written by changing the logical variable PSIFMT to .FALSE. in the appropriate driver routine.

13.7 Log-derivative matrices

If ILDSVU > 0, the log-derivative matrix at $R_{\max}$ is output on unit ILDSVU. This may be required for other programs, such as those to implement MQDT [67, 68].

Since this file is designed to be processed by other programs, a full description of what is written to it is given here. This description requires significant understanding of internal structures and variable names, and should be read only by expert users who need it. It has not yet been generalised to handle multiple EFVs.

In brief, the output on unit ILDSVU has the structure:

1. Global header
2. Global vector
   Looping over number of propagations
   (a) Propagation header
   (b) Propagation external fields
   (c) Propagation vectors
   Looping over number of matrices
   i. Matrix data

These various parts contain the following variables/values:

Global header contains

```
LABEL, ITYPE, NSTATE, NLEVEL, NQN, NNRG, NFIELD, URED, IPROGM,
NDGVL, NCONST, NRSQ, IBOUND, ISVEFV, NEFV,
(EFVNAM(IEFV), EFVUNT(IEFV), IEFV = 1, NEFV),
```

Global vectors contains

`JSTATE(NSTATE,NQN), ELEVEL(NLEVEL), ENERGY(NNRG)`

Propagation header contains

`JTOT, INRG, EN, IEXCH, WT, M, NCH, ERED, RMLMDA`

Propagation external fields contains

`(EFV(IEFV), IEFV = MIN(1,ISVEFV), MAX(NEFV,ISVEFV))`

Propagation vectors contains

`JSINDX, L, EINT`, which all have length `NCH`.

Matrix data contains

`MATCODE` (For future expansion: currently a large negative integer)

`RMID`, `Y` (`Y` is the `NCH` by `NCH` log-derivative matrix)

13.8 Coupling matrices

This option is designed for cases where there are many expansion terms contributing to the interaction potential (and/or other operators) and it requires excessive memory to store them internally. If `IVLU` > 0, the coupling matrices are written to unit `IVLU` and the program reads them back in one at a time when constructing the interaction matrix. This saves memory at the expense of disc I/O, so is generally worthwhile only when the available memory is otherwise insufficient.

Chapter 14

Example input and output files

We have provided a selection of example input files and their associated outputs, to give examples of program features and to allow users to verify that their program build is operating correctly. These are in subdirectory `examples/input`, with the corresponding output files in `examples/output`.

The example calculations are intended to be illustrative, and in some cases use basis sets or propagation parameters that are not fully converged.

Instructions for building executables to run the example calculations are given in Section 15.11.

14.1 Examples for MOLSCAT

14.1.1 All available propagators for MOLSCAT

input file: `molscat-all_propagators.input`
executable: `molscat-basic`

`molscat-all_propagators.input` contains input data for the same model of collisions between an atom and a linear rigid rotor as was used for the basic example in section 3.8.1. The radial potential coefficients are provided in the input file and consist of a Lennard-Jones 12-6 potential for $\lambda = 0$ and a dispersion-like R^{-6} form for $\lambda = 2$. The program carries out full close-coupling (`ITYPE` $= 1$) calculations for a single partial wave and total parity and prints the resulting S matrix. The calculation is repeated using combinations of short-range and long-range propagators that exercise every propagation method available in MOLSCAT (though not every possible combination).

14.1.2 All available coupling approximations (using ITYP = 2)

input file: `molscat-all_iadds.input`
executable: `molscat-basic`

`molscat-all_iadds.input` contains input data for a similar model system, extended this time to include vibrations of the linear rotor (`ITYP` = 2). The radial potential coefficients are again provided in the input file, and all consist of inverse-power functions of R. The LDMD/AIRY hybrid propagation scheme is used. MOLSCAT first performs close-coupling calculations (`ITYPE` = 2) and then repeats the calculation using every decoupling approximation available (`ITYPE` = 12, 22, 32, 102).

14.1.3 Locating and characterising a quasibound state (Feshbach resonance) for Ar-HF

input file: `molscat-Ar_HF.input`
executable: `molscat-Rg_HX`

`molscat-Ar_HF.input` demonstrates the procedure for locating a quasibound state, which appears as a narrow resonance in the S-matrix eigenphase sum as a function of energy. The procedure is described in section 9.9. It performs calculations on the H6(4,3,2) potential of Hutson [69] for the ground ($v = 0$) vibrational state of HF, using the LDMD propagator. The first calculation sets `NNRG` = −10; it first solves the coupled equations at 5 energies reasonably close to the resonance (but actually over 1000 widths away) and uses the resulting eigenphase sums to estimate the resonance position. It then chooses another 5 energies around the estimated resonance position, and this time finds that they span the resonance (which is between points 2 and 3 of the second set of 5). The formula used for estimating resonance positions is valid only far from resonance, so it reports that the second set of points cannot safely be used to locate the resonance energy.

The procedure used to estimate the resonance position in this example amplifies any tiny differences between computers due to finite-precision arithmetic. The second set of 5 energies is commonly significantly different on different computers; this does not indicate an error.

The second calculation characterises the resonance using the algorithm of Frye and Hutson [12]. The initial three energies for this calculation have been chosen based on the estimated resonance position and width from the first 5 energies above. The algorithm converges quickly on the resonance position and gives accurate results for the resonance energy and width.

14.1.4 Line-shape cross sections for Ar + CO_2

input file: `molscat-Ar_CO2.input`
executable: `molscat-Rg_CO2`

`molscat-Ar_CO2.input` performs close-coupling calculations of line-shape cross sections for the S(10) Raman line of CO_2 in Ar, using the single-repulsion potential of Hutson *et al.* [70] with the LDMD propagator at short range and the AIRY propagator at long range. The calculations are at a kinetic energy of 200 cm^{-1} and the total energies are calculated internally. The program prints cross sections accumulated up to the current value of `JTOT`; the convergence of the partial-wave sum may be compared with Fig. 2 of ref. [71].

14.1.5 Line-shape cross sections for Ar + H_2

input file: `molscat-Ar_H2.input`
executable: `molscat-Rg_H2`
also required: `data/h2even.dat`

`molscat-Ar_H2.input` calculates pure rotational Raman line widths and shifts across a shape resonance at a collision energy near 14 cm^{-1}. It uses the BC_3(6,8) interaction potential of Le Roy and Carley [72], evaluated for H_2 states $(j, v) = (0, 0)$, (2,0) and (4,0) using H_2 matrix elements in the file `data/h2even.dat`. The line-shape calculations require S matrices evaluated at the same *kinetic* energy for different rotational states of H_2; the program treats the input energies as kinetic energies and generates the total energies required. The results may be compared with Figure 2(a) of ref. [27].

14.1.6 ITYP = 3: Cross sections for rigid rotor + rigid rotor collisions

input file: `molscat-ityp3.input`
executable: `molscat-H2_H2`

`molscat-ityp3.input` contains input data for collisions between pairs of H_2 molecules. The interaction potential is that of Zarur and Rabitz [73].

The first 4 calculations are for para-H_2 (even j) colliding with ortho-H_2 (odd j). MOLSCAT calculates elastic and state-to-state inelastic cross sections. Collisions do not transfer molecules between even and odd j, so identical-particle symmetry is not included. Contributions from different partial waves are accumulated until the partial-wave sums are converged within the limits set by the input data. Two calculations are performed, first with close-coupling calculations and then with the coupled-states approximation. Each calculation is done twice; once with the radial potential coefficients supplied explicitly, and once with the unexpanded potential supplied and expanded by quadrature by the program. The results illustrate the equivalence of the two methods.

The final calculation is for para-H_2 colliding with para-H_2. In this case identical-particle symmetry is important, and is included.

All calculations use the LDMD/AIRY hybrid propagation scheme.

14.1.7 ITYP = 5: Cross sections for atom + symmetric top collisions, with automated testing of propagator convergence

input file: `molscat-ityp5.input`
executable: `molscat-basic`

`molscat-ityp5.input` contains input data for atom + symmetric top collisions between He and ortho-NH_3, taking account of the tunnelling splitting of NH_3. It uses a simple analytical interaction potential and the LDMD/AIRY hybrid propagation scheme. The input file uses ISYM(3) = 1 to select rotational functions of E symmetry and ISYM(4) = 1 to specify that the H nuclei are fermions. The first four calculations are for a single partial wave and exercise the convergence-testing code in MOLSCAT, testing the convergence with respect to step size

(chosen in two ways), and with respect to the start point and end point of the propagation. The final two calculations carry out full cross-section calculations, using converged values for the propagation variables, first with close-coupling calculations and then with the coupled-states approximation.

14.1.8 ITYP = 6: Cross sections for atom + spherical top collisions

input file: `molscat-ityp6.input`
executable: `molscat-Ar_CH4`

`molscat-ityp6.input` contains input data for atom + spherical top collisions between Ar and CH_4, using the interaction potential of Buck *et al.* [74]. The ground-state rotational constants and the tetrahedral centrifugal distortion constant d_t are specified in the input file and the program uses them to calculate properly symmetrised spherical-top wavefunctions. The input file selects CH_4 rotor functions of A symmetry by setting ISYM to 224, as described on p. 49. The cross sections use the automatic total angular momentum option JTOTU = 99999 with a convergence tolerance (OTOL) of 0.0001 to give well-converged inelastic cross sections, but the diagonal convergence tolerance DTOL is set to 10.0 so that the partial-wave sum terminates before the elastic cross sections are converged. The results may be compared with Table VI of ref. [75], although they do not agree exactly because the results in the paper are averaged over the experimental distribution of collision energies.

14.1.9 ITYP = 8: Atom-surface scattering

input file: `molscat-ityp8.input`
executable: `molscat-basic`

`molscat-ityp8.input` contains input data for diffractive scattering (ITYPE = 8) of He from solid LiF, using the model potential of Wolken [28]. It uses the LDMD propagator at two energies.

14.1.10 ITYP = 9: Cross sections for Mg + NH in a magnetic field

input file: `molscat-Mg_NH.input`
executable: `molscat-Mg_NH`
also required: `data/pot-Mg_NH.data`

`molscat-Mg_NH.input` contains input data for cold collisions of NH with Mg in a magnetic field. It uses the plug-in basis-set suite described in section 18.1, for a $^3\Sigma$ diatom colliding with a structureless atom. Radial potential coefficients are provided by a (VINIT/VSTAR) routine that applies RKHS interpolation to the interaction potential of Soldán *et al.* [76]. The coupled-channel equations are solved using the LDMD/AIRY hybrid propagation scheme (IPROPS = 6, IPROPL = 9).

The basis-set suite implements two different forms of the monomer Hamiltonian, including and excluding the off-diagonal matrix elements of the spin-spin operator. The input file specifies

runs with both of these (IBSFLG = 2 and 1 respectively). The approximation does not actually produce any saving in computer time in this case.

The input file requests calculations at kinetic energies of 1, 10 and 100 mK above the $n = 0$, $j = 1$, $m_j = 1$ threshold of NH by using LOGNRG = .TRUE. to select a logarithmically increasing energy set as described in section 6.2.

These calculations are similar to (a subset of) those of Wallis *et al.* [77], although the test run uses a smaller basis set than ref. [77]. Convergence at 100 mK requires inclusion of incoming partial waves up to $L = 3$, which requires values of $M_{\rm tot}$ from -2 to 4 for incoming $m_j = 1$. This is represented in the input file with JTOTL = -2, JTOTU = 4.

MOLSCAT can accumulate cross sections from calculations for different values of $M_{\rm tot}$ and total parity for a single EFV set. The first part of the test run, with IBSFLG = 2, illustrates the scheme used for identification of levels for systems with non-diagonal Hamiltonians, where not all threshold channels may be known at the point where the first partial cross sections are calculated.

14.1.11 ITYP = 9: Characterisation of magnetically tunable Feshbach resonances and quasibound states and calculation of effective range for ^{85}Rb + ^{85}Rb

input file: molscat-Rb2.input
executable: molscat-Rb2

molscat-Rb2.input contains input data for low-energy ^{85}Rb + ^{85}Rb collisions in a magnetic field, using the plug-in basis-set suite described in section 18.2 and the potential of Strauss *et al.* [78], implemented with the VINIT/VSTAR routine described in section 19.5. This is the same system used for the basic resonance scan described in section 3.10.1. All the calculations use the LDMD/AIRY hybrid propagation scheme (IPROPS = 6, IPROPL = 9).

This test run characterises 4 different low-energy Feshbach resonances as a function of magnetic field, using the characterisation algorithms described by Frye and Hutson [12, 13]. The first resonance is in purely elastic scattering in the lowest (aa) scattering channel, so produces a pole in the scattering length as a function of magnetic field. The second and third resonances occur in collisions at excited thresholds, where weak inelastic scattering is possible and the pole is replaced by an oscillation [14]. The fourth is subject to strong background inelasticity. After these calculations, it characterises a quasibound state just below the ee threshold at 155 G, using the algorithm of Frye and Hutson [12]. Finally, a further calculation obtains the effective range across the strong resonance observed in the aa channel near 850 G, described in section 3.10.1, from scans across the resonance at energies of 100 and 200 nK.

The basis-set suite for this interaction requires information about the hyperfine properties of the atoms in an additional namelist block named &BASIS9, as described in section 18.2. The potential expansion comprises 3 terms: the singlet and triplet interaction potentials, and the spin-spin dipolar term, which is modelled as in Eq. 18.10, with coefficients specified in namelist &POTL. The radial coefficient is scaled by $-E_{\rm h}\alpha^2$ internally, so that the items given in &POTL are A(1) = $-g_S^2(a_0/\mathrm{RM})^3/4$, A(2) = $-A$ and E(1) = $-\beta$ RM$/a_0$.

14.1.12 Simple 2-channel scattering problem

input file: `molscat-2chan-LJ.input`
executable: `molscat-basic`

The programs are designed to handle matrix elements that are products of potential coefficients $v^\Lambda(R)$ and angular momentum factors $\mathcal{V}^\Lambda_{ij}$, as in Eq. 2.7 (and more explicitly Eq. 17.1). The built-in coupling cases implement the angular momentum factors for common cases in atomic and molecular scattering, and plug-in basis-set suites can be used to implement other cases in a very general way.

Users may wish to implement a simple N-channel coupled-channel with the matrix elements supplied directly as coefficients $v^\Lambda(R)$. This can be done by choosing an interaction type where each $\mathcal{V}^\Lambda$ is a matrix containing all zeroes except for a single element (for a particular i, j) that is 1. The simplest way to set this up is to use ITYPE = 2 (diatomic vibrotor + atom), with a basis set made up only of functions with $j = 0$. For example, for a 2×2 matrix, set NLEVEL = 2 and JLEVEL = 0, 1, 0, 2 in namelist &BASIS. If the diagonal matrix elements have parts that are independent of R, it is best to set them in ELEVEL(1) and ELEVEL(2), rather than coding them as part of the interaction potential, since the programs then recognise them as threshold energies. If the R-independent diagonal parts are both zero, it is necessary to set the elements of ELEVEL to some very small value (say 1.D-30) so that the programs do not complain that they cannot calculate threshold energies.

In namelist &POTL, specify the potential coefficient that is placed in each matrix element by setting MXLAM = 3 and LAMBDA = 0, 1, 1, 0, 2, 2, 0, 1, 2.

Finally, either supply the required matrix elements as data in &POTL or write a VINIT/VSTAR routine that returns the matrix elements $\langle 1|V(R)|1\rangle$, $\langle 2|V(R)|2\rangle$ and $\langle 1|V(R)|2\rangle$ when called with I = 1, 2 and 3 respectively.

To generate a set of coupled equations without additional centrifugal terms, run the programs with JTOTL = JTOTU = 0 and IBFIX = 2 in namelist &INPUT.

The input file `molscat-2chan-LJ.input` demonstrates this for a 2-channel scattering problem. The units of length and energy are set to Å and cm^{-1}, and URED is set to 20 m_u. The threshold energies for the 2 channels are set in ELEVEL as 0 and 100 cm^{-1}. Both diagonal matrix elements are Lennard-Jones 12-6 potentials,

$$V(R) = 4\epsilon \left[\left(\frac{R}{\sigma_0}\right)^{-12} - \left(\frac{R}{\sigma_0}\right)^{-6} \right].$$

These are constructed to have well depth ϵ and $V(\sigma_0) = 0$; here, $\epsilon = 100$ cm^{-1} and $\sigma_0 = 1$ Å. The two channels are coupled by another Lennard-Jones potential whose strength is a factor of 10 smaller than the diagonal potentials.

The file specifies two runs. The first run is for scattering at an energy of 200 cm^{-1}, where both channels are open, and produces a 2×2 S matrix. The second run is at an energy where channel 1 is open but channel 2 is closed, and calculates the eigenphase sum for a scan over a Feshbach resonance just below 42.84 cm^{-1}. As will be seen in example 14.2.10, the corresponding bound system has a bound state at -58.3 cm^{-1}, so this resonance may

be identified as due to a similar bound state in the upper channel, shifted by the 100 cm^{-1} difference in threshold energies and slightly shifted further by the coupling.

Large sets of coupled equations should *not* be constructed in this way, but instead by writing a basis-set suite to generate the coupling matrices in terms of quantum numbers and a smaller set of radial potential coefficients.

14.2 Examples for BOUND

14.2.1 All available propagators for bound-state calculations

input file: `bound-all_propagators.input`
executable: `bound-basic`

`bound-all_propagators.input` performs close-coupling calculations on the bound states of a simple model of a complex formed between an atom and a linear rigid rotor. The radial potential coefficients are provided in the input data file and consist of a Lennard-Jones 12-6 potential for $\lambda = 0$ and a dispersion-like R^{-6} form for $\lambda = 2$. The calculation is repeated using combinations of short-range and long-range propagators that exercise every propagation method available in BOUND (though not every possible combination). The calculation is done twice for the LDMD/AIRY combination; once with $R_{\mathrm{mid}} < R_{\mathrm{match}}$ and once with $R_{\mathrm{mid}} > R_{\mathrm{match}}$. The calculation which uses just the LDMD propagator employs a different step length for the inwards propagation. This input file should produce the same results regardless of which BOUND executable is used.

14.2.2 Bound states of Ar-HCl with expectation values and wavefunction

input file: `bound-Ar_HCl.input`
executable: `bound-Rg_HX`

`bound-Ar_HCl.input` performs calculations on the states of Ar-HCl bound by more than 80 cm^{-1}, using the H6(4,3,0) potential of Hutson [79] and the LDMD propagator, for total angular momentum $J_{\mathrm{tot}} = 0$ and 1 and both parities. The first run does close-coupling calculations. The second run does calculations in the helicity decoupling approximation, and in addition calculates expectation values $\langle P_2(\cos\theta)\rangle$ and $\langle 1/R^2\rangle$ for all the states. The results may be compared with Table IV of ref. [79]. The third run calculates the wavefunction for the first bound state identified in the first run. The wavefunction is written to unit 109; the resulting file is included as `bound-Ar_HCl.wavefunction` in `examples/output`. The components may be plotted with any standard plotting package.

14.2.3 Bound states of Ar-CO_2 with Richardson extrapolation

input file: `bound-Ar_CO2.input`
executable: `bound-Rg_CO2`

`bound-Ar_CO2.input` performs close-coupling calculations on the ground and first vibra-

tionally excited state of Ar-CO_2, using the split-repulsion potential of Hutson *et al.* [70] and the LDJ propagator, for total angular momentum $J_{\rm tot} = 0$. The results may be compared with Table IV of ref. [70].

It first calculates the ground-state energy using a fairly large (unconverged) step size of 0.03 Å. It then repeats the calculation with an even larger step size, and extrapolates to zero step size using Richardson h^4 extrapolation.

14.2.4 Bound states of Ar-H_2

input file:	`bound-Ar_H2.input`
executable:	`bound-Rg_H2`
also required:	`data/h2even.dat`

`bound-Ar_H2.input` performs close-coupling calculations on the ground state of Ar-H_2 with H_2 in its $v = 1$, $j = 1$ state, for total angular momentum $J_{\rm tot} = 1$ and even total parity ($j + L$ even). For this parity there is no allowed $j = 0$ channel, so the state is bound except for vibrational predissociation to form H_2 ($v = 0$) [80], which is not taken into account by BOUND. The run uses the LDMD propagator and the TT3(6,8) potential of Le Roy and Hutson [81], evaluated for H_2 states $(j, v) = (0, 0)$, (2,0) and (4,0) using H_2 matrix elements in the file `data/h2even.dat`.

BOUND first calculates the ground-state energy using a fairly large (unconverged) step size of 0.04 Å. It then repeats the calculation with an even larger step size, and extrapolates to zero step size using Richardson h^4 extrapolation.

14.2.5 Bound states of H_2-H_2 (ortho-para)

input file:	`bound-ityp3.input`
executable:	`bound-H2_H2`

`bound-ityp3.input` performs close-coupling calculations on bound states of H_2-H_2 with one para-H_2 molecule (even j) and one ortho-H_2 molecule (odd j). It uses the LDMD propagator. The interaction potential is that of Zarur and Rabitz [73]. The states are bound by less than 2 cm^{-1} (below the j=0 + j=1 threshold).

14.2.6 Bound states of He-NH_3

input file:	`bound-ityp5.input`
executable:	`bound-basic`

`bound-ityp5.input` performs close-coupling calculations on bound states of He-NH_3, taking account of the tunnelling splitting of NH_3, using a simple analytical interaction potential and the LDMD propagator. The input file selects rotational functions of E symmetry by setting `ISYM(3)` to 1 and specifies that the H nuclei are fermions by setting `ISYM(4)` to 1.

14.2.7 Bound states of Ar-CH_4

input file: `bound-Ar_CH4.input`
executable: `bound-Ar_CH4`

`bound-Ar_CH4.input` performs close-coupling calculations on bound states of Ar-CH_4, using `ITYPE` = 6, which can also handle complexes of asymmetric tops. It uses the interaction potential of Buck *et al.* [74]. It uses the LDMD propagator. CH_4 is a spherical top, and the input file selects rotor functions of F (T) symmetry by setting `ISYM` to 177, as described on p. 49. It may be noted that the particular one of each set of degenerate rotor functions that has even k is numerically arbitrary, so the values of the quantum number τ generated by the program may vary for different computers or compilers. The results may be compared with Table II of ref. [26].

14.2.8 Bound-state energies of the hydrogen atom

input file: `bound-hydrogen.input`
executable: `bound-basic`

`bound-hydrogen.input` carries out single-channel bound-state calculations on the hydrogen atom, and demonstrates how to handle calculations in atomic units. It sets `MUNIT` to the electron mass in Daltons, `RUNIT` to the Bohr radius in Å and `EUNITS` = 7 to select input energies in hartrees. It uses the general-purpose `POTENL` to set up a simple Coulomb potential, with `EPSIL` set to the hartree in cm^{-1}, so that the potential is handled in atomic units. It uses `ITYPE` = 1 with `JMAX` = 0 to generate a simple single-channel problem. Note that `ROTI(1)` is set to the dummy value `1.0`; this value is not used because `JMAX` = 0, but it prevents the program terminating prematurely.

The wavefunction at the origin is of the form r^{l+1}, so its log-derivative is infinite at the origin. This is the default for locally closed channels, but is specified explicitly for the locally open $l = 0$ channel.

Because `JMAX` = 0, the orbital angular momentum l is equal to `JTOT`. `JTOT` = 0 produces ns levels at energies of $-1/(2n^2)$ for $n = 1, 2, ...$, while `JTOT` = 1 produces np levels starting at $n = 2$.

14.2.9 Bound-state energies of Mg-NH at specified magnetic fields

input file: `bound-Mg_NH.input`
executable: `bound-Mg_NH`
also required: `data/pot-Mg_NH.data`

`bound-Mg_NH.input` locates the bound states of Mg-NH at specified magnetic fields. It uses a plug-in basis-set suite for a $^3\Sigma$ diatom colliding with a structureless atom. Radial potential coefficients are obtained by RKHS interpolation of the potential points of Soldán *et al.* [76]. The coupled equations are solved using the LDMD/AIRY hybrid propagation scheme.

The run locates a single bound state at four different magnetic fields from 370 G to 385 G, from which it may be inferred that the state crosses threshold near 387 G.

14.2.10 Simple 2-channel bound-state problem

input file: `bound-2chan-LJ.input`
executable: `bound-basic`

This example solves for the bound states of the simple 2-channel problem described in example 14.1.12. It has two channels asymptotically separated by 100 cm^{-1} and each described by a Lennard-Jones potential with a well 100 cm^{-1} deep. The two channels are coupled by a Lennard-Jones potential whose strength is a factor of 10 smaller.

BOUND locates two bound states, at -58.2 cm^{-1} and -12.1 cm^{-1}. The coupling is fairly weak compared to the separation of the channels, so these are only weakly perturbed from the corresponding single-channel levels (not located here) at -57.7 cm^{-1} and -11.9 cm^{-1}.

14.3 Examples for FIELD

14.3.1 Bound states of Mg-NH as a function of magnetic field

input file: `field-Mg_NH.input`
executable: `field-Mg_NH`
also required: `data/pot-Mg_NH.data`

`field-Mg_NH.input` locates magnetic fields in the range 0 to 400 G at which bound states exist for specific energies relative to the lowest scattering threshold of Mg + NH in a magnetic field. It uses the same basis-set suite and interaction potential as in section 14.2.9. The coupled equations are solved using the LDMD/AIRY hybrid propagation scheme. The AIRY propagator uses a power-law step size (`TOLHIL` = 0) in place of the default adaptive step-size algorithm; this choice eliminates step-size noise resulting from the large range of the eigenvalues of the potential matrix, as described on p. 85, and significantly improves the convergence on magnetic fields at which bound states exist.

The run locates the same level as in section 14.2.9 at energies of 0, 20 and 40 MHz $\times$ h below threshold, and shows that it crosses threshold near 387.28 G.

14.3.2 Locating threshold crossings for $^{85}Rb_2$

input file: `field-basic_Rb2.input`
executable: `field-Rb2`

`field-basic_Rb2.input` locates magnetic fields where bound states cross the lowest scattering threshold for $^{85}Rb_2$. These are the fields at which zero-energy Feshbach resonances exist. It uses a plug-in basis-set suite for a pair of alkali-metal atoms in a magnetic field, including hyperfine interactions. It uses the potential of Strauss *et al.* [78], implemented with potential coefficients incorporated in the executable. The coupled equations are solved using the LDMD/AIRY hybrid propagation scheme.

This is the same example as in section 3.10.2.

14.3.3 Bound states of $^{85}Rb_2$ as a function of magnetic field

input file: `field-Rb2.input`
executable: `field-Rb2`

`field-Rb2.input` locates bound states of $^{85}Rb_2$ as a function of magnetic field, using the same potential and basis-set suite as in section 3.10.2. The calculation locates the magnetic fields (in the range 750 to 850 G) at which bound states exist with binding energies of 225, 175, 125, 75 and 25 MHz below the lowest threshold. There are, however, two bound states that these calculations fail to find, as they run almost parallel to the threshold, at about 140 and 220 MHz below it. To locate these bound states, one would need to do a calculation using BOUND.

Chapter 15

Installing and testing the programs

15.1 Supplied files

This distribution is supplied as a tarred zipped file `2019molscat.tar.gz`, that contains:

- the full program documentation in pdf format;
- a directory `source_code` containing
  - the Fortran source code;
  - a makefile (`Makefile`) that can build the executables needed for the example calculations described in sections 3.8, 3.9 and 3.10 and in chapter 14;
- a directory `examples` containing
  - a sub-directory `input` containing input files for the example calculations described below;
  - a sub-directory `output` containing the corresponding output files;
- a directory `data` containing auxiliary data files for some potential routines used in the example calculations;
- a plain-text file `README` that gives information on changes that may be needed to adapt the makefile to a specific target computer.
- a plain-text file `COPYING` that contains the text of the GNU General Public License, Version 3.

To demonstrate how to handle pointwise potential coefficients (which often result from electronic structure calculations) by interpolation, we have included examples that use RKHS interpolation on such a data set for Mg+NH.

15.2 Program language

MOLSCAT, BOUND and FIELD are written in near-standard Fortran 77 with some Fortran 90 features, such as the use of modules. Most of the code is in files with `.f` extensions that use Fortran 77 spacing conventions. A small number of routines are in files with `.f90` extensions that use Fortran 90 spacing conventions.

The programs have been tested with current versions of `gfortran`, `ifort` and `pgf90`.

15.3 Main routine

The main routine is common to all the programs. It does not do any processing; it simply declares storage and calls the relevant version of `DRIVER` (`mol.driver.f`, `bd.driver.f` or `fld.driver.f`) to do all the work.

15.4 Integer length

The programs obtain working storage by partitioning an array of type `DOUBLE PRECISION`. On most machines, `DOUBLE PRECISION` values occupy 8 bytes each, while integers occupy only 4 bytes, so it is possible to pack 2 integers into each 8-byte element. The variable `NIPR`, set in subroutine `DRIVER`, must be equal to the number of integers that may be packed into 8 bytes. `NIPR` should be 2 on most machines.

15.5 Date, time and CPU time routines

The programs obtain the date and time of a run (for output in the header) by calls to routines `GDATE` and `GTIME` and information on the CPU time taken by calls to subroutine `GCLOCK`. The distribution provides versions of these routines that call the Fortran 90 utility routines `date_and_time` and `cpu_time`.

15.6 Linear algebra routines

The programs use LAPACK linear algebra routines wherever possible.

If possible, run the programs using LAPACK routines that are optimised for your particular computer. However, if this is not possible, Fortran versions of the LAPACK routines may be obtained from the Netlib repository (www.netlib.org).

The LAPACK routines use BLAS (basic linear algebra subroutines) as much as possible. BLAS level 1, level 2 and level 3 routines exist. Use BLAS routines optimised for your particular computer if possible. However, if no optimised routines are available, Fortran versions may be obtained from the Netlib page at www.netlib.org/blas.html.

Any user who implements new options in any of the programs should perform matrix operations by calls to the routines described below, both for ease of maintenance and to simplify the creation of efficient executables for other computers.

Linear algebra routines supplied with MOLSCAT:

`DGEMUL` Matrix multiplication
`DGESV` Solve linear equations
`SYMINV` Invert symmetric matrix
`DIAGVL` Diagonalise symmetric matrix without eigenvectors
`DIAGVC` Diagonalise symmetric matrix with eigenvectors

The programs also call BLAS routines such as `DAXPY`, `DDOT` etc. in many places.

15.6.1 Matrix multiplication

The programs call `DGEMUL`. This was originally a routine from the IBM ESSL library. In version 2020.0, `DGEMUL` calls the BLAS routine `DGEMM`. The Fortran 90 subroutine `ytrans` (`ytrans.f90`) uses matrix operators such as `matmul` rather than calling `DGEMUL` because that makes the code more readable. Efficiency is not usually an issue for `ytrans` since it is called only a few times per run.

15.6.2 Symmetric matrix inversion

Symmetric matrix inversion is a key operation that dominates the time taken by some propagators, so its efficiency is important. The programs call `SYMINV`. The version of `SYMINV` included in version 2020.0 calls the LAPACK routines `DSYTRF` and `DSYTRI` to carry out the inversion for matrix sizes above 30. For smaller matrices it calls a pure Fortran routine. The threshold for switching between the two could be changed if desired for optimum efficiency on a specific machine.

Note that the programs really do require matrix inversion, despite the usual advice to use linear equation solvers instead. This is because the propagators save information from one step to the next, and this advantage is lost if the problem is formulated in terms of linear equation solvers.

15.6.3 Linear equation solver

The programs call the LAPACK routine `DGESV` directly. The speed of this routine is not critical for most propagators.

15.6.4 Eigenvalues and eigenvectors of symmetric matrices

These routines are important for propagators 3, 4, 7, and 9 (RMAT, VIVS, LDMA and AIRY). The programs call diagonalisers via routines `DIAGVC` (for eigenvalues and eigenvectors) and

DIAGVL (for eigenvalues alone). These routines both make a call to the LAPACK routine DSYEVR.

The versions of DIAGVC, DIAGVL and SYMINL supplied use allocatable arrays to supply workspace for the LAPACK routines that they call.

15.7 File handling

The programs adhere to the Fortran 77 standard in their use of READ and WRITE statements (including direct access files).

The OPEN statements do not use FILE = 'fname' parameters and where necessary the user must provide files with the naming convention for their system. For example, most Linux systems use filename fort.NN if unit NN is opened.

15.8 Dimensions of variably sized arrays

The main routine for the programs declares one large array, X, which is held in COMMON /MEMORY/. The specification of this common block is:

```
COMMON /MEMORY/ MX, IXNEXT, NIPR, IDUMMY, X

DOUBLE PRECISION :: X(MX)
INTEGER          :: MX, IXNEXT, NIPR
```

The value of MX is set in the main routine, which calls the relevant driver routine to run one of the programs. The relevant DRIVER and other routines then partition the array X according to the size of the problem being tackled, and specific elements are passed into subroutines to act as the first element of arrays. Thus, very few of the arrays used internally by the programs are explicitly dimensioned, and it is seldom necessary for users to concern themselves with array dimensioning.

If a program finds at run time that the array X is not big enough, it (in most cases) terminates with the message

```
CHKSTR.  CANNOT PROVIDE REQUESTED STORAGE.
```

It is then usually sufficient to modify the main routine to increase the parameter MXDIM to specify a larger array X and recompile, as described in section 15.13. Note that the error message specifies the storage required for the current allocation request, which may not be sufficient for the entire program run.

If a user wishes to use part of the array X in their own code, they must place any new arrays beyond the block currently used (which is specified by the variable IXNEXT in common block /MEMORY/. To check that the space needed is available, they should (prior to using the arrays)

set IXNEXT to its previous value plus the total needed for additional arrays, and then make the subroutine call CALL CHKSTR(NUSED) with NUSED initialised to 0. If insufficient space is available in the rray X, CHKSTR stops the program and the prints the message above. Space in the array X may be released by resetting the value of IXNEXT to the value it had before the temporary assignment.

There are a few internal arrays that are handled differently. These are mostly arrays that are either input as data or are in modules, and thus cannot be flexibly dimensioned. A small subset of these (see below) are dimensioned using parameters contained in modules sizes and efvs; their dimensions can be changed with minimal effort if required. The rest are sufficiently large that they should suffice for all situations.

15.9 Dimensions of fixed-size arrays

All three programs use a module sizes, which contains parameters used to set dimensions of certain arrays. These parameters are:

- MXFLD which limits the number of EFV values that can be entered in &INPUT and the length of loops over EFV sets;
- MXNRG which limits the number of energies that can be looped over;
- MXNODE which limits the number of states that can be searched for in a single instance of a loop over the energy (for BOUND) or one EFV set (for FIELD);
- MXLN which limits the number of sets of lines for pressure broadening calculations;
- MXLOC which limits the number of propagations that may be used to locate each bound state or field-dependent resonance;
- MXJLVL which limits the size of the array JLEVEL in module basis_data; see section 17.12.1 below;
- MXELVL which limits the size of the array ELEVEL in module basis_data; see section 17.12.1 below;
- MXROTS which limits the size of the array ROTI in module basis_data; see section 17.12.1 below;
- MXSYMS which limits the size of the arrays ISYM and ISYM2 in module basis_data; see section 17.12.1 below;
- MXOMEG which limits the sizes of the arrays VCONST and NEXTMS in module potential; see section 17.12.2 below;
- MXLMDA which limits the size of the array LAMBDA in module potential; see section 17.12.2 below;
- MXANG which limits the size of the array COSANG in module angles; see section 5.3.2.

The dimension of arrays that are used for information about EFVs depend on a parameter `MXEFV`, which is set in module `efvs`.

15.10 `COMMON` blocks

The programs use a number of `COMMON` blocks internally, and the names of these should be avoided when naming `COMMON` blocks in subroutines that link with the distributed code. A brief description of these common blocks is given below.

`ASSVAR` is used in very old code. Passes variable between `DASIZE` and `PRBR`.

`BCCTRL` contains variables used for setting boundary conditions.

`CNTROL` contains a character variable `CDRIVE`, which can take the values `M`, `B` or `F`. This indicates whether the executable is for MOLSCAT, BOUND or FIELD.

`DERIVS` contains a logical variable which controls whether derivatives of the interaction potential are calculated numerically or analytically (when required).

`EIGSUM` contains eigenphase sums used for estimating the position of a nearby (energy) resonance.

`EXPVAL` contains variables relevant to the calculation of expectation values.

`IOCHAN` contains unit numbers for `IPSI`, `IPSISC` and `IWAVSC`, together with some variables controlling how a wavefunction is written on unit `IPSI`

`IOUTCM` contains variables to pass information between `IOSOUT` and `IOSBIN`.

`LATSYM` contains logical variables for surface scattering calculations.

`LDVVCM` contains variables for the VIVS propagator.

`MEMORY` is described in section 15.8.

`NPOT` contains the variable `NVLP`, which is used to set `NHAM` when `IVLFL` > 0.

`POPT` contains variables to control level of printing for the VIVS propagator.

`PRBASE` contains variables relevant to calculations of line-shape cross sections.

`PRPSCR` contains variables relevant to use of scratch files for propagation segments.

`RADIAL` contains variables controlling the propagation segments.

`VLFLAG` contains the variable `IVLFL`, which indicates whether the array `P` is indexed using the array `IV` to allow the array `VL` to be smaller. This facility is currently used only for `ITYP` $= 2$, 7 and 8.

`VLSAVE` contains unit number for storage of the `VL` array.

`WKBCOM` contains variables relevant to the Gauss-Mehler quadrature performed for WKB integration.

15.11 Compiling and linking the programs

We have provided a basic makefile in the file `Makefile`. This is designed to compile the programs and build the executables required to run the examples in Chapter 14. It has been tested with GNU `make`, which is the default version of `make` on Linux and OS X. Modifications may be needed for other versions of `make`.

The basic form of the command is
`make` *executable-name*
where *executable-name* is one of the executables listed in Chapter 14, such as `molscat-basic`, `bound-basic` or `field-basic`.

The makefile may need some minor modifications for a specific installation. If this is done, we strongly recommend leaving the supplied version in `Makefile` unchanged, and making a copy for modification in either `makefile` or `GNUmakefile`. If either of these files exists, the GNU `make` command will use it in preference to `Makefile`.

The supplied makefile sets up `gfortran` as the compiler in the variable `Compiler`, but this can be changed if required. We have tested the makefile with the `gfortran`, `pgf90` and `ifort` compilers. The `LIBS` variable (which is currently unset) should if possible contain the names of optimised LAPACK and BLAS libraries that are on the library path, each prefixed by `-l`; on many systems, setting `LIBS = -llapack -lblas` will work. If this is not possible, the individual library routines must be downloaded as described in section 15.6, and the names of the object (`.o`) files included in the variable `LIBUTILS`.

The reason for leaving the supplied version of `Makefile` unchanged is that future versions of the programs may require changes in it. If the original file is changed, the `git pull` command used to update the programs from `https://github.com` (Section 1.3) may fail because it is unable to update the file.

If `Makefile` is updated by `git pull`, you will need to make a new active copy based on the updated version, and transfer your changes to it before using it to recreate the executables.

15.11.1 If `make` gets tangled

Various unexpected occurrences can cause `make` to become confused. The most common examples involve Fortran module files (`.mod`), which appear to be handled inconsistently by some versions of `make`. If unexpected error messages appear, it is often sufficient to delete all files with names ending `.mod` and `.o` and recompile from scratch using
`make` *executable-name*

15.12 Testing the installation

After building the executables, it is highly desirable to validate the programs by running all the examples described in Chapter 14 and verifying that the programs give output essentially identical to that in the files supplied.

The supplied output files were obtained from executables compiled with `gfortran` and run on a machine with `x86_64` architecture. Different compilers may produce values that are formatted slightly differently. Some quantities are output at close to machine precision, so may have slightly different values with different compilers or on different architectures. In addition, convergence procedures may take slightly different steps, even if they converge to essentially identical points. It is therefore necessary to exercise some judgement in deciding whether results differ *significantly* from the test output.

15.13 Adding a new executable to the makefile

Many users will wish to create new executables that include their own routines for the interaction potential and/or the basis set. To do this, it is necessary to extend the supplied makefile to provide rules to create the new executables.

15.13.1 Fundamentals and terminology of `make`

The following instructions need at least a basic understanding of the contents of makefiles. A short glossary of terminology may be useful:

- A *target* is a file that `make` has a *rule* to create.
- A *rule* is a set of commands to create a *target* from a list of its *dependencies*.
- A *dependency* is a file that is required (and must be up-to-date) to create the *target*.

If a target already exists, `make` recreates it only if one or more of its dependencies has a time stamp newer than the target (i.e., has changed since the last time the target was created).

Dependencies are applied recursively; each dependency may have its own dependencies, and will itself be recreated if necessary.

15.13.2 Editing the makefile

Make a copy of `Makefile` in either `makefile` or `GNUmakefile`, as described in Section 15.11. Edit the copy rather than the original.

A new executable is specified by adding lines to the makefile to specify its dependencies and a rule for creating it. The dependencies for a new executable must include a main routine as described below, one of the dependency lists `CORE_MOL`, `CORE_BND` and `CORE_FLD`, for MOLSCAT, BOUND and FIELD, respectively, and the list `REPLACEABLE`, which lists object files for subroutines that expert users may wish to replace for special purposes.

Additional dependency lists as described below are needed to specify the object files required for

1. the interaction potential;
2. the plug-in basis-set suite.

The dependency lists of this type in the supplied version of `Makefile` all begin with `POT-` or `BASE9-` as appropriate and serve as examples for constructing your own dependency lists. We

recommend adding any new dependency lists immediately after the supplied ones.

If you have previously edited the makefile for a program version earlier than 2020.0, you will need to repeat the edits using the new names for the dependency lists.

Main routine

The main routine is common to all three programs. Its only function is to set the size of the array `X` that is used to hold variably dimensioned arrays as described in section 15.8. The main routine supplied with this distribution is `main1M.f`, in which the array `X` is dimensioned at 1 million 8-byte words, which is sufficient for most problems with up to about 200 channels. However, larger problems can require much larger arrays. To change the dimension, simply make a copy of `main1M.f` in which the parameter `MXDIM` is changed as required, and include the corresponding object file in the rule for the executable in place of `main1M.o`.

Dependency list for the interaction potential

- If the potential is supplied entirely within the input data file, use the existing list `POT-BASIC`;
- If the potential is supplied using one of the other dependency lists already set up in `Makefile`, use that list;
- Otherwise, set up your own dependency list, using a line of the form
  potential-dependency-list-name = list of dependencies
  where *potential-dependency-list-name* is a name of your choosing.
  - If you wish to use the general-purpose version of `POTENL`, the list should contain `$(POTENL-GP)` followed by the unqualified names of the object files containing the versions of `VSTAR` and/or `VRTP` that you wish to include. In most cases at least one of these will be a routine you have supplied. If either of `VSTAR` and `VRTP` is *not* used by your implementation, use `vstar-dummy.o` or `vrtp-dummy.o` for that one. The dependency list should also contain the names of any other object files that contain routines called uniquely by your versions of `VSTAR` and/or `VRTP`.
  - If you do not wish to use the general-purpose version of `POTENL`, the list should contain the unqualified name of the object file containing your version of `POTENL`, followed by the names of any other object files that contain routines called uniquely by it.

Dependency list for the plug-in basis-set suite

- If you do not wish to use a plug-in basis-set suite, use the existing list `BASE9-UNUSED`.
- Otherwise, set up your own dependency list, using a line of the form
  basis-set-dependency-list-name = list of dependencies
  where *basis-set-dependency-list-name* is a name of your choosing. The list should contain

the unqualified name(s) of the object file(s) containing your plug-in basis-set suite (see chapter 17 for a list of routines). If your basis-set suite does not include specific versions of the additional routines `THRSH9`, `EFV9` or `DEGEN9`, include the supplied dummy versions of these routines (`thrsh9-dummy.o`, `efv9-dummy.o` or `degen9-nondegenerate.o` respectively). If you wish to use the general-purpose version of `POTENL`, you must also specify an object file containing the `POTIN9` routine. If you do not require a specific one, use the dummy version (`potin9-example.o`).

Name for the new executable

Add the unqualified name for the new executable to the list of targets in `USER-PROGS`, which is empty in the supplied `Makefile`.

Rule to create the new executable

Add a rule to create the new executable. We recommend that this is placed at the end of the list of supplied executables. Use the rules already in `Makefile` as a template to build your own rule. On the first line, give the relative filename of the new executable, followed by a colon, followed by a list of dependencies. These may be split over several lines as in the example below; each line that is continued ends with a backslash (\) character.

```
$(EXECDIR)/executable-name:  $(addprefix $(OBJDIR)/,mainsize.o \
                                                    $(CORE_type-of-executable) \
                                                    $(replaceable-dependency-list-name) \
                                                    $(basis-set-dependency-list-name) \
                                                    $(potential-dependency-list-name)
```

where the items in italic must be replaced by your own strings. *replaceable-dependency-list-name* should be `REPLACEABLE` unless you are an expert user who has replaced one of the special-purpose routines in it. On the line immediately following, put a `TAB` character followed by the rule for creating the target. The following formula is sufficient in most situations:

```
<TAB>$(LINK.f) $^ $(LIBS) -o $@
```

Module requirements

If any of your routines uses one or more of the supplied modules, the name of your object file should be added to the list of files that must be re-made if the module is changed or recompiled. The names of these lists all end in `_DEPS`, with the first part of the name being a fairly obvious reference to the module concerned.

15.13.3 Creating the executable

Finally, create the new executable as described in section 15.11 with the command
`make` *executable-name*

15.14 Values of fundamental constants

All three programs use values of fundamental physical constants and derived quantities in the module `physical_constants`. The current values are the 2018 CODATA recommended values; a module containing the 2014 values is provided in `physical_constants_2014.f`. The programs print a message stating the date of the values used.

All the values set in `physical_constants` are parameters and most have long, non-standard names. This is to ensure that they are clearly differentiated from the Fortran standard variable names used in the rest of the code.

To use the values in `physical_constants` in your own code, insert the line
`USE physical_constants`
immediately after the routine declaration statement.

To use the 2014 values in place of the 2018 ones, make a copy of `Makefile` in either `makefile` or `GNUmakefile`, as described in Section 15.11. Then edit the copy to remove the hash (`#`) character from the beginning of the following lines:

```
#physical_constants.mod: physical_constants_2014.f \
#                        $(OBJDIR)/physical_constants_2014.o
#       @true
```

and

```
#MODULES =   physical_constants_2014.o efvs_module.o sizes_module.o \
#            potential_module.o basis_module.o angles_module.o
```

Finally, delete any executable with the same name with the command
`rm` *executable-name*
and recreate it as described in section 15.11 with the command
`make` *executable-name*

Chapter 16

Plug-in potential routine (POTENL)

As described in chapter 5, the programs internally require an expansion of the interaction potential in a set of orthogonal functions of the internal coordinates,

$$V(R, \xi_{\mathrm{intl}}) = \sum_{\Lambda=1}^{\mathtt{MXLAM}} v_\Lambda(R) V^\Lambda(\xi_{\mathrm{intl}}). \tag{16.1}$$

For most interaction potentials, this can be handled using the general-purpose version of subroutine POTENL, which may call VINIT/VSTAR to provide the radial potential coefficients, or perform an integration by quadrature to obtain the coefficients from an unexpanded potential provided by routine VRTP.

In rare cases where these mechanisms are inconvenient or inefficient, the user may supply a complete routine POTENL to replace the general-purpose version. This section describes the specification of this routine.

16.1 Specification of POTENL subroutine

In each run, POTENL is called once for initialisation purposes, and on this call may read any data necessary to specify the interaction potential. It returns information about the terms present in the potential expansion. Subsequently, POTENL is called many times during each propagation to evaluate the radial potential coefficients $v_\Lambda(R)$ for particular interparticle distances R.

The syntax of a call to POTENL is

```
CALL POTENL(IC, MXLMB, LAMBDA, RR, P, ITYPE, IPRINT)

DOUBLE PRECISION, INTENT(OUT)   :: RR, P(MXLMB)
INTEGER,          INTENT(OUT)   :: LAMBDA(NLABV,MXLMB)
INTEGER,          INTENT(INOUT) :: MXLMB
INTEGER,          INTENT(IN)    :: IC, ITYPE, IPRINT
```

The array LAMBDA specifies MXLAM sets of NLABV integers. Each set identifies a term Λ in the expansion (Eq. 16.1).

The array P specifies MXLAM radial potential coefficients $v_\Lambda(R)$, in the same order as the elements of LAMBDA.

The arrays LAMBDA and P should be dimensioned as LAMBDA(NLABV,1) and P(1) to switch off Fortran array bound checking, since MXLAM is not known at the time of an initialisation call.

There are two basic types of call to POTENL;

Initialisation: POTENL is called once with IC $= -1$, before any other calls to it, to allow it to read any necessary data and set up parameters for later use.

Evaluation: At subsequent calls to POTENL, IC is 0, 1 or 2 and the routine must evaluate the radial potential coefficients or their radial derivatives. As described below, radial derivatives are not really essential.

The specification of POTENL for initialisation and evaluation calls is described separately.

16.1.1 Initialisation

MXLMB: On entry, MXLMB specifies the maximum dimension that has been externally provided for the LAMBDA array. This value may be (and is, in the provided general-purpose version) used to check for array bound errors.

On exit, MXLMB must be set equal to MXLAM, which specifies the number of distinct terms in the expansion of the interaction potential (i.e., the dimension of the P array that is returned by subsequent calls to POTENL).

LAMBDA: On exit, the LAMBDA array must contain indices specifying the potential terms to be used. Although it is externally a one-dimensional array, it is conceptually two-dimensional for some interaction types, and may be handled explicitly as a two-dimensional array in POTENL by declaring it as LAMBDA(NLABV,1) and declaring NLABV as a parameter. Each element (or column) of LAMBDA corresponds to an element of the P array returned by subsequent calls to POTENL. The programs do not require that the symmetry terms be supplied in any particular order, but just that the ith column of the LAMBDA array should correspond to the ith element of the P array.

The explicit form of the expansions is described for the built-in interaction types in sections 5.1.1 to 5.1.6; the value of NLABV can be obtained by counting the number of labels that comprise Λ in the table on page 57. For ITYPE $= 9$, NLABV is set in routine SET9, described in section 17.6.

RR: On exit, RR must specify the length units RM that are used in subsequent calls to POTENL and are used for most quantities with dimensions of length output by the programs (except cross sections). RM must be returned in Å. It is often convenient to set RR = 1.0D0 in the initialisation call to POTENL, and to handle everything in Å thereafter.

P: On exit, P(1) must specify the energy scaling factor EPSIL (expressed in cm^{-1}) to be used internally by the programs, and subsequent calls to POTENL must return energies in units of EPSIL. It may be convenient to set EPSIL = 1.0D0 in the initialisation call to POTENL, and to handle everything in cm^{-1} thereafter. The value given to EPSIL does *not* affect the interpretation of energy parameters input in namelist &INPUT and &BASIS, or output energies other than the interaction potential.

ITYPE: On entry, ITYPE is the interaction type.

If POTENL is coded specifically for a particular value of ITYPE, it should check that the correct value has been passed, as a precaution against the accidental use of the wrong executable version. The value of this parameter should not be changed by POTENL.

IPRINT: used to control the quantity of output produced by POTENL.

16.1.2 Evaluation (IC=0, 1 or 2)

For an evaluation call to POTENL, only the IC, MXLMB, RR, P and IPRINT arguments are passed. LAMBDA and ITYP do *not* contain the values they were given in the initialisation call, so copies of these must be stored internally in POTENL if they are needed in an evaluation call.*

IC = 0 evaluate the radial potential coefficients v_Λ(RR) and return them in P

IC = 1 evaluate dv_Λ/dR at R = RR and return them in P

IC = 2 evaluate d^2v_Λ/dR^2 at R = RR and return them in P

RR: The interparticle distance at which the potential is to be evaluated, in units of RM; see the RR argument for an initialisation call to POTENL above.

P: On exit, P must contain the array of radial potential coefficients (or their derivatives) at distance RR, in the order specified earlier by the LAMBDA array returned by the initialisation call to POTENL. The P array must be returned in units of EPSIL$\times$(RR)$^{-\text{IC}}$; see the discussion of the initialisation call above).

Calls to POTENL with IC = 1 or 2 occur only:

- for the VIVS propagator if IVP, IVPP, ISHIFT or IDIAG is set;
- for the LDMA propagator at high print levels (in order to calculate the nonadiabatic couplings).

Even in these cases, there is an option (controlled by logical variable NUMDER in namelist &INPUT) that allows the derivatives to be evaluated numerically without making calls to POTENL with IC = 1 or 2. It is thus not altogether necessary for POTENL to cope with IC = 1 and 2 calls, but it should at least trap an attempt to call it this way and print an error message.

*In the general-purpose version of POTENL, this is achieved by (i) naming the LAMBDA array LAM internally and saving the elements read in from the namelist as LAMBDA internally in module potential whilst a copy is passed out as the dummy array LAM; (ii) saving MOD(ITYPE,10) as an internal variable ITYP.

Chapter 17

Plug-in basis-set suites

The programs provide a facility to construct and solve sets of coupled equations that are different from those for the built-in interaction types. This chapter gives the information needed to write a suite of plug-in subroutines to do this. It may be skipped by readers who wish only to run existing codes.

The routines described in this chapter are called *only* if ITYPE = 9. They are not needed for any of the built-in interaction types (ITYP = 1 to 8).

The distribution includes a skeleton version of a plug-in basis-set suite (`base9-skeleton.f`), which will halt if called, but contains comments that can be used as guidance for writing a new suite. The distribution also includes two plug-in basis-set suites, described in chapter 18, which may be used as examples by programmers of new routines.

17.1 Components of a basis-set suite

When coding a new plug-in basis-set suite, the programmer must always provide the following routines:

routine	task to perform	see section
BAS9IN	Read any data needed to specify the basis set, in addition to quantities read in &BASIS.	17.5
SET9	Set up the lists of pair levels and pair states.	17.6
BASE9	Set up the basis set for the current JTOT and IBLOCK	17.7
POTIN9	Choose the type of potential expansion to be used if it is one of the built-in types, or set up the variables needed for the potential expansion from scratch if not.	17.8
CPL9	Calculate the coupling matrices of the expansion functions used for the interaction potential in the current basis set. If $H_{\rm intl}$ and/or $\hat{L}^2$ is non-diagonal, matrix elements of the operators used to expand them are also required.	17.9

Many older basis-set suites code most or all of these routines as entry points to BAS9IN, so

that variables in `SAVE` statements are common to all of them. However, for new basis-set suites it is preferable to code the routines as separate subroutines, with variables to be shared in a Fortran module.

In addition, the programmer *may* need to write the following:

routine	task to perform	see section
`DEGEN9`	Calculate denominators for degeneracy-averaged cross sections (if necessary)	17.10.1
`THRSH9`	Calculate threshold energies from monomer quantum numbers (if necessary)	17.10.2
`EFV9`	Transform the input external field variables (EFVs) into the components used for coupling matrices (if necessary)	17.10.3

Dummy versions of `DEGEN9`, `THRSH9` and `EFV9` are supplied in `degen9-nondegenerate.f`, `thrsh9-dummy.f` and `efv9-dummy.f`. These must be linked in unless a bespoke version has been programmed.

Routines in plug-in basis-set suites need access to variables that are not in their argument lists. These are contained in a few modules, described in section 17.12: module `basis_data` contains variables related to basis sets and pair levels; module `potential` contains variables related to internal Hamiltonians and extra operators; module `efvs` contains variables related to EFVs.

The routines listed above are described in sections 17.5 to 17.9 in the order in which they are called.

17.2 Diagonal or non-diagonal asymptotic Hamiltonian

The first choice to make is the basis set to use. There are often many possible basis sets for a given problem. Different basis sets give equivalent results when they are complete, but they often offer different opportunities for approximations that involve restricting the basis set. In addition, BOUND and FIELD produce wavefunctions that are expanded in the basis set, and these may be easier to interpret for one choice of basis set than another.

The programs handle two different types of basis set:

1. Basis sets in which $H_{\rm intl}$ and $\hat{L}^2$ are diagonal;
2. Basis sets in which $H_{\rm intl}$ and/or $\hat{L}^2$ are non-diagonal.

These are be described as diagonal and non-diagonal basis sets in this chapter, although the interaction potential is almost always non-diagonal (as otherwise single-channel rather than coupled-channel calculations suffice). For a given problem, non-diagonal basis sets are often simpler and easier to program, though the resulting output is sometimes more complicated to interpret. Problems involving EFVs usually require non-diagonal basis sets, since the same basis set seldom diagonalises $H_{\rm intl}$ at different values of the EFVs.

Diagonal and non-diagonal basis sets use different subsets of the internal variables, and require significantly different programming as described below. A non-diagonal basis set is indicated

by returning a positive value of NCONST and/or NRSQ from BAS9IN as described below.

17.3 Interpretation of external loop variables

The loop structure in MOLSCAT and BOUND is conceptually

```
Read &INPUT
Read &BASE
Call BAS9IN (usually reads &BASE9)
Call SET9
Call DEGEN9 (MOLSCAT only, some input options only)
Initialise potential; read &POTL and call POTIN9
DO JTOT = JTOTL, JTOTU, JSTEP
  DO IBLOCK = 1, NBLOCK
    Call BASE9
    Call CPL9
    DO IFIELD = 1, NFIELD (external fields)
      Call THRSH9 (only for some input options for threshold energies)
      Call EFV9 (only for some input options for magnetic fields)
      DO INRG = 1, NNRG (energies)
        Propagations, with many evaluation calls to POTENL
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

For FIELD, the loops over energies and external fields are reversed.

The programmer is free to use JTOT and IBLOCK for any purpose desired. For field-free calculations, it is natural to use JTOT for the total angular momentum $J_{\rm tot}$ and IBLOCK for any additional symmetries in the Hamiltonian (total parity, body-fixed K, etc.). However, for calculations in a magnetic field, $J_{\rm tot}$ is not conserved. For a single field (or parallel fields), however, its projection $M_{\rm tot}$ onto the field axis may be conserved; in this case the variable JTOT is conveniently used for $M_{\rm tot}$.

The programs hold quantum numbers in integer variables and arrays. For systems with half-integer spins, it it often convenient to store *doubled* quantum numbers. Most quantum number values are processed *only* within the basis-set suite (though they may be printed), so doubling their values causes no problems for processing in the remainder of the programs. The exceptions to this are

- The array L, described in section 17.7.
- The loop variable JTOT, which may be used in different ways depending on the value of JHALF, which should be set within the basis-set suite. The only operation outside the basis-set suite that needs explicit knowledge of how JTOT is used is the evaluation of degeneracy-averaged cross sections from Eq. 2.12, which contains a factor of $2J_{\rm tot}+1$:

`JHALF` $= 1$ indicates that `JTOT` is an undoubled total angular momentum $J_{\rm tot}$;

`JHALF` $= 2$ indicates that `JTOT` is a doubled total angular momentum, $2J_{\rm tot}$;

`JHALF` $= 0$ indicates that `JTOT` is not a total angular momentum, and omits the factor $(2J_{\rm tot}+1)$ from the cross section.

17.4 Calculating the interaction matrix

At each step of a propagation, the propagators require the interaction matrix defined by Eq. 2.5. This may be written

$$W_{ij}(R) = \frac{2\mu}{\hbar^2}\left(\sum_{\Lambda} v_\Lambda(R)\mathcal{V}^\Lambda_{ij} + \sum_{\Omega} h_\Omega \mathcal{H}^\Omega_{ij}\right) + \sum_{\Upsilon} \mathcal{L}^\Upsilon_{ij}/R^2. \tag{17.1}$$

All the coupling matrices $\boldsymbol{\mathcal{V}}^\Lambda$, $\boldsymbol{\mathcal{H}}^\Omega$ and $\boldsymbol{\mathcal{L}}^\Upsilon$ are calculated prior to the propagation by `CPL9`.

For both diagonal and non-diagonal basis sets, the potential coupling matrices $\boldsymbol{\mathcal{V}}^\Lambda$ are stored in the array `VL`. During the propagation, the R-dependent coupling coefficients $v_\Lambda(R)$ are supplied by `POTENL`.

For non-diagonal basis sets, `NCONST` coupling matrices $\boldsymbol{\mathcal{H}}^\Omega$ and/or `NRSQ` centrifugal matrices $\boldsymbol{\mathcal{L}}^\Upsilon$ are also stored in the array `VL`.

For diagonal basis sets, the coupling matrices $\boldsymbol{\mathcal{H}}^\Omega$ must also be diagonal. Any contributions that are independent of EFVs may be calculated by `SET9` and included in the elements of the array `ELEVEL`. Alternatively (and necessarily for terms that depend on EFVs) `NDGVL` blocks of diagonal elements may be calculated by `CPL9` and stored in the array `DGVL`. The pair energies are obtained from

$$E_{{\rm intl},i} = E^{\rm field-free}_{{\rm intl},i} + \sum_{\Omega} h_\Omega \mathcal{H}^\Omega_{ii}. \tag{17.2}$$

The R-independent coupling coefficients h_Ω are either supplied by `BAS9IN` or, if they depend on EFVs, generated internally. Both EFV-dependent and EFV-independent coefficients are stored in the array `VCONST`, with `NCONST` elements for non-diagonal basis sets or `NDGVL` elements for diagonal basis sets.

If there is only one centrifugal matrix $\boldsymbol{\mathcal{L}}^\Upsilon$, and it is diagonal, with elements of the form $\mathcal{L}_{ij} = L_i(L_i+1)$, the integers L_i may be returned in the array `L` by `BASE9`. If the matrix is diagonal but its elements are not of the form $L_i(L_i+1)$, `IBOUND` may be set to 1 by `BAS9IN` and the diagonal elements of $\hat{L}^2$ returned in the array `CENT` by `CPL9`. If the centrifugal matrices are non-diagonal, `NRSQ` must be set greater than zero and the full matrices $\boldsymbol{\mathcal{L}}^\Upsilon$ returned by `CPL9`.

17.5 Routine `BAS9IN`

```
SUBROUTINE BAS9IN(PRTP, IBOUND, IPRINT)
```

```
USE potential

CHARACTER(32), INTENT(OUT)   :: PRTP
INTEGER,       INTENT(INOUT) :: IBOUND
INTEGER,       INTENT(IN)    :: IPRINT
```

BAS9IN is an initialisation routine. It is called by BASIN, and so is called only once in a particular run. It has access to most of the quantities read in &BASIS through the module basis_data, but can read additional information if required. This has usually been done using a namelist block &BASE9, but that is not compulsory. The corresponding input data must be included in the input file between &BASIS and &POTL. They commonly include limits on the pair levels to be included and values of spectroscopic constants for the interacting particles. Some of these quantities are needed by other routines in the basis-set suite; this may be achieved by placing the shared variables in a Fortran module.

The variable IPRINT gives the print level for the current calculation and may be used to control how much is printed by BASIN.

BAS9IN must return the following quantities:

PRTP: character string containing a brief description of the interaction type, which is printed in the output.

IBOUND: if NRSQ $= 0$, IBOUND $= 0$ (the default) indicates that centrifugal energies are to be calculated from values assigned to the array L by BASE9. IBOUND > 0 indicates that the array L should not be used and centrifugal energies are to be calculated from values in the array CENT returned from CPL9. If NRSQ $\neq 0$, neither L nor CENT is used to calculate centrifugal energies and IBOUND is not used.

It is usually appropriate to use IBOUND $= 0$ if the matrix elements of the centrifugal potential are diagonal and of the form $\hbar^2 L(L+1)/(2\mu R^2)$ with integer L, and IBOUND $\neq 0$ otherwise.

BAS9IN must set the following quantities that are included in module potential:

NCONST: set to 0 if $H_{\rm intl}$ is diagonal in the basis set. For non-diagonal basis sets, the number of terms in the expansion (2.8) of $H_{\rm intl}$, including terms used for EFVs (see below).

NDGVL: for diagonal basis sets, the number of diagonal terms in the expansion (2.8) of $H_{\rm intl}$, including terms used for EFVs (see below).

VCONST: array of R-independent coefficients for the terms in the expansion of $H_{\rm intl}$, as described in sections 17.4 and 17.9. Values should be set by the end of POTIN9 unless they depend on EFVs. The product of an element of VCONST and an element of VL or DGVL should be in units of cm^{-1}. VCONST must be set to 0 for any operator not to be included in $H_{\rm intl}$. Required only if NCONST > 0 or NDGVL > 0.

NRSQ: set to 0 if centrifugal potentials are diagonal and are specified either by the array CENT or calculated from values in the L array. Otherwise, the number of terms in the expansion of the operator $\hat{L}^2$. NRSQ > 1 is not currently supported.

If (and only if) extra operators are required to resolve degeneracies, their structure should be defined here:

`NEXTRA`: number of extra R-independent operators.

`NEXTMS`: array giving the number of coupling matrices for each extra R-independent operator.

If the elements of the array `VL` are to be indexed using the array `IV`, as described in section 17.9, `BAS9IN` should set the variable `IVLFL` to 1 in common block `VLFLAG`. If `IVLFL` = 1, either `BAS9IN` or `POTIN9` should set `NVLP` (in common block `NPOT`) to the number of blocks of the array `VL` used for the interaction potential.

A number of internal variables in module `potential` are set, based on these variables, after the initialisation call to `POTENL`:

If `IVLFL` = 0 (the default),

$$\texttt{NHAM} = \texttt{MXLAM} + \texttt{NCONST} + \texttt{NRSQ}. \tag{17.3}$$

If `IVLFL` = 1,

$$\texttt{NHAM} = \texttt{NVLP} + \texttt{NCONST} + \texttt{NRSQ} \tag{17.4}$$

(but `IVLFL` = 1 is currently implemented only for `NCONST` = `NRSQ` = 0).

In either case,

$$\texttt{NEXBLK} = \sum_{i=1}^{\texttt{NEXTRA}} \texttt{NEXTMS}(i), \tag{17.5}$$

$$\texttt{NVLBLK} = \texttt{NHAM} + \texttt{NEXBLK}. \tag{17.6}$$

BOUND includes a facility to calculate expectation values using a finite-difference method [66], as described in section 10.5. To use this, the operator concerned must be part of $H_{\rm intl}$. If it is not naturally part of $H_{\rm intl}$, it may be added as an additional term (and included in `NCONST`), with the corresponding coefficient in `VCONST` set to zero.

If external fields are to be included in the calculation, `BAS9IN` must also set quantities in module `efvs` that describe them. Access to these quantities must be gained by using module `efvs`. The quantities that must be set are:

`NEFV`: the number of EFVs.

`EFVNAM`: array of character strings describing the EFVs, with maximum lengths specified in module `efvs`.

`EFVUNT`: array of short character strings giving names of units for the EFVs, with maximum lengths specified in module `efvs`.

MAPEFV: a positive value specifies the index of the first EFV in the VCONST array; a negative value indicates a non-linear mapping between the EFVs and the corresponding elements of VCONST. The options are described in section 17.10.3.

BAS9IN may also set the variable ITPSUB in module efvs. This is written by MOLSCAT on unit ISAVEU so that an external program can identify the basis-set suite that produced the results.

The namelist item DEGTOL is used as a threshold for degeneracy for both $H_{\rm intl}$ and the extra operators. It is treated as an energy and so is scaled according to EUNITS or EUNIT. Programmers of extra operators should ensure that the operators are scaled such that DEGTOL is an appropriate threshold for degeneracy (which usually implies that they should have eigenvalues that span a range between 1 and 10^3).

17.6 Routine SET9

```
SUBROUTINE SET9(LEVIN, EIN, NSTATE, JSTATE, NQN, QNAME, NBLOCK, NLABV,
IPRINT)

INTEGER,      INTENT(OUT) :: NQN, NBLOCK, NLABV, NSTATE, JSTATE(*)
CHARACTER(8), INTENT(OUT) :: QNAME(10)

LOGICAL,      INTENT(IN)  :: LEVIN, EIN
INTEGER,      INTENT(IN)  :: IPRINT
```

This routine is called by BASIN shortly after the call to BAS9IN.

SET9 must always return values for the following quantities:

NQN: one greater than the number of quantum labels used to specify a pair state.

QNAME: array of names of the quantum labels used to specify a pair state.

NBLOCK: the number of independent symmetry blocks for each value of JTOT.

NLABV: the number of labels used to specify a term in the potential expansion.

The logical variable LEVIN indicates whether the array JLEVEL was supplied explicitly in &BASIS; however, it is quite unlikely that this mechanism would be required for a new basis-set suite, and it is usually sufficient to print an error message and stop if LEVIN is .TRUE. The logical variable EIN is .TRUE. either if the corresponding energies were given in &BASIS in the array ELEVEL, or if values from which they can be calculated were input in the array ROTI.

The variable IPRINT gives the print level for the current calculation and may be used to control how much is printed by SET9.

For diagonal basis sets, SET9 must set the following quantities in module basis_data, unless they were input as explicit arrays in &BASE.

NLEVEL: the number of pair levels (not pair states). Pair levels are used for diagonal basis sets to label state-to-state cross sections (and are distinct from pair *states*, as described in section 4.2.1.

JLEVEL: array of quantum labels that specify pair levels. Can be given in &BASIS, but more commonly calculated from limits on quantum numbers. Each set of quantum labels must be unique.

ELEVEL: array of energies of pair levels, corresponding to JLEVEL. Can be given in &BASIS, but more commonly calculated from input spectroscopic parameters.

For non-diagonal basis sets, the programmer *may* if desired use NLEVEL and the arrays JLEVEL and ELEVEL, but they are not used outside the basis-set suite for non-diagonal basis sets, so this is optional.

For both diagonal and non-diagonal basis sets, SET9 must return:

NSTATE: the number of pair states.

JSTATE: array of labels for the pair states, arranged as though in an array of dimension (NSTATE, NQN). For diagonal basis sets, the last column must contain the index of the element of the array ELEVEL that contains the energy of the pair state. For non-diagonal basis sets, the last column is not used and can be left unset.

If the number of quantum numbers needed to specify a pair level (as opposed to a pair state) is less than NQN − 1, and the array MONQN will be used to specify a reference energy, SET9 must set the variable NJLQN9 in module basis_data.

17.7 Routine BASE9

```
SUBROUTINE BASE9(LCOUNT, N, JTOT, IBLOCK, JSTATE, NSTATE, NQN, JSINDX, L,   &
                 IPRINT)

INTEGER, INTENT(INOUT) :: N

INTEGER, INTENT(OUT)   :: JSINDX(N), L(N)

LOGICAL, INTENT(IN)    :: LCOUNT
INTEGER, INTENT(IN)    :: JTOT, IBLOCK, NSTATE, NQN, JSTATE(NSTATE,NQN),    &
                          IPRINT
```

Routine BASE9 is called by BASE to set up the basis set for the current combination of JTOT and IBLOCK. Since the calling program does not initially know the size of the basis set, BASE9 is called twice: first to *count* the basis functions, and subsequently to set up the basis functions themselves.

When LCOUNT is .TRUE. on entry, BASE9 must count the required basis functions and return the number of them in N. In this case, no space has yet been allocated for the array JSINDX and L, and so BASE9 must *not* assign values in the arrays.

When LCOUNT is .FALSE. on entry, BASE9 must set up the JSINDX and L arrays. Each function in the basis set is specified by an element in each of the JSINDX and L arrays:

JSINDX(i): a pointer to a pair state in the JSTATE array.

L(i): a value of L that, when combined with the pair quantum numbers indexed by the corresponding element of JSINDX, specifies a function in the basis set. Used only if IBOUND $= 0$ and NRSQ $= 0$, unless the programmer chooses to use it in CPL9.

N may be smaller or larger than NSTATE, and (for a single JTOT and IBLOCK) JSINDX may reference only a subset of the pair states.

IPRINT gives the print level for the current calculation and may be used to control how much is printed by BASE9.

17.8 Routine POTIN9

```
SUBROUTINE POTIN9(ITYPP, LAM, MXLAM, NPTS, NDIM, XPT, XWT, MXPT,            &
                  IVMIN, IVMAX, L1MAX, L2MAX, MXLMB, XFN, MX, IXFAC)

INTEGER,          INTENT(INOUT) :: ITYPP, MXLAM
INTEGER,          INTENT(OUT)   :: LAM(1)

! the quantities below are only utilised if quadrature is to be used
INTEGER,          INTENT(INOUT) :: NDIM, NPTS(NDIM), IXFAC, MX

DOUBLE PRECISION, INTENT(INOUT) :: XFN(*)
DOUBLE PRECISION, INTENT(OUT)   :: XPT(MXPT,NDIM), XWT(MXPT,NDIM)

INTEGER,          INTENT(IN)    :: MXPT, IVMIN, IVMAX, L1MAX, L2MAX, MXLMB
```

POTIN9 is called by the general-purpose version of POTENL during an initialisation call (when IC $= -1$).

In most cases, it is sufficient for POTIN9 to select a value of ITYP that has the desired expansion of the interaction potential and return that value in ITYPP. POTENL then uses the logic for that value of ITYP to determine MXLAM, construct the LAMBDA array and (if LVRTP = .TRUE.) evaluate radial potential coefficients by quadrature.

If a value of ITYPP other than 9 is returned, none of the other arguments in the calling sequence need to be used.

In rare cases, none of the built-in potential types is suitable. In such cases, POTIN9 must return ITYPP $= 9$ and also:

MXLAM: the number of potential expansion terms. Passed into and out from POTIN9.

LAM: the labels for the potential expansion terms, in an array of dimension (NLABV,MXLAM).

To do this, POTIN9 may use IVMIN, IVMAX, L1MAX, L2MAX from the argument list, which are the values input in &POTL, and/or values in module basis_data. If additional quantities are needed, POTIN9 may read its own input data, commonly in a namelist block &POTL9. The LAM array returned from POTIN9 is passed into CPL9 and so can be used in construction of the VL array. If other routines also need to know which expansion terms are included in the current calculation, they may obtain them from the array LAMBDA in module potential.

Note that NLABV, the number of integers needed to label each term in the potential expansion, is set by SET9 rather than POTIN9. It is always required, even if POTIN9 sets a value of ITYPP other than 9.

If ITYPP = 9 and radial potential coefficients are to be obtained by quadrature (LVRTP = .TRUE.), POTIN9 must also set up the sets of quadrature points, weights and functions to be used. The quadrature is written (for 2 dimensions, but easily extended to more)

$$v_\Lambda(R) = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} w_{i_1}^{(1)} w_{i_2}^{(2)} f^\Lambda \left(\xi_{i_1}^{(1)}, \xi_{i_2}^{(2)}\right) V \left(R, \xi_{i_1}^{(1)}, \xi_{i_2}^{(2)}\right), \tag{17.7}$$

where $\xi_i^{(d)}$ and $w_i^{(d)}$ are the n_d points and weights required for the quadrature over the functions $f^{\lambda_d}(\xi^{(d)})$; f^Λ is constructed from the product of the functions f^{λ_d} for all the labels λ_d that make up Λ. For this, POTIN9 must return

NDIM: number of dimensions of which quadratures are to be used.

NPTS: array of numbers of points used for each quadrature.

XPT: array of sequential sets of the points used for the NDIM quadratures.

XWT: corresponding array of weights.

XFN: array of quadrature functions $f^\Lambda(\xi_{i_1}^{(1)}, \xi_{i_2}^{(2)}, \ldots)$.

IXFAC: the array of quadrature functions must be stored at the end of the XFN array (which is actually the X array, held in common block MEMORY), from positions IXFAC+1 to the value of MX on entry. The array is of dimension NFUN = MXLAM × NPTS(1) × ... × NPTS(NDIM), with the innermost loop over MXLAM. On exit, IXFAC must be set to the value of MX on entry − NFUN.

MX: on entry, MX is the available space in the X array. POTIN9 must reset MX to MX − NFUN.

17.9 Routine CPL9

```
SUBROUTINE CPL9(N, IBLOCK, NHAM, LAM, MXLAM, NSTATE, JSTATE, JSINDX, L,      &
                JTOT, VL, IV, CENT, DGVL, IBOUND, IEXCH, IPRINT)
```

```
DOUBLE PRECISION, INTENT(OUT) :: VL(NVLBLK,N*(N+1)/2), CENT(N),              &
                                 DGVL(N,NDGVL)
INTEGER,          INTENT(OUT) :: IV(NVLBLK,N*(N+1)/2)

INTEGER,          INTENT(IN)  :: N, IBLOCK, NHAM, LAM, MXLAM, NSTATE,        &
                                 JSTATE(NSTATE,*), JSINDX(N), L(N), JTOT,    &
                                 IBOUND, IEXCH, IPRINT
```

Routine CPL9 is called by BASE, once for each JTOT and symmetry block IBLOCK, to calculate the elements of the coupling matrices, which are returned in the arrays VL and DGVL.

Two different structures are implemented for the array VL, controlled by the variable IVLFL in common block VLFLAG:

1. The usual case is for IVLFL = 0. The first MXLAM blocks of VL must return the coupling matrices $\boldsymbol{\mathcal{V}}^{\Lambda}$ for the interaction potential.

2. The second case, with IVLFL = 1, allows the size of the array VL to be reduced for some interaction types where only a few potential expansion coefficients $v_\Lambda(R)$ contribute to each element $W_{ij}(R)$ of the interaction matrix. NVLP $\leq$ MXLAM is the maximum number of potential coefficients that contribute to any element.* The first NVLP blocks of VL must return the coupling matrices, and an additional array IV, of the same dimension as VL, specifies how they are used. Each element IV(I) indicates that the corresponding element VL(I) should subsequently be multiplied by the potential expansion coefficient with index IV(I).

For non-diagonal basis sets, subsequent blocks of VL must return coupling matrices for the NCONST R-independent operators $\boldsymbol{\mathcal{H}}^{\Omega}$ and the NRSQ centrifugal operators $\boldsymbol{\mathcal{L}}^{\Upsilon}$. Such operators are currently implemented only for IVLFL = 0, but this description is written to allow future generalisation.†

The elements of the array VL (and optionally IV) must be arranged in the order corresponding to the loop structure:

```
IRC = 0
DO ICOL = 1, N
  DO IROW = 1, ICOL
    IRC = IRC + 1
    DO IPOTL = 1, NVLBLK
```

*The mechanism with IVLFL = 1 is used by some of the built-in coupling cases, specifically ITYP = 2, 7 and 8. It has not been tested with a plug-in basis-set suite, but it should be compatible.

†Routine YTRANS would require extension to implement IVLFL = 1 for non-diagonal basis sets.

The total number of coupling matrices used for calculating the interaction matrix is NHAM, given by Eq. 17.3 or 17.4.

If extra operators are required to resolve threshold degeneracies, as described in section 9.2, their coupling matrices must be returned as NEXBLK extra blocks of the array VL. The total number of coupling matrices stored in the VL array is then NVLBLK = NHAM + NEXBLK. The variables NEXBLK and NVLBLK are defined by Eqs. 17.5 and 17.6 and are stored in module potential.

```
      VL(IPOTL, IRC) = the (IROW,ICOL)-th element of the IPOTL-th coupling matrix
      IF (IVLFL.GT.0) IV(IPOTL, IRC) = the index of the potential expansion coefficient
    ENDDO
  ENDDO
ENDDO
```

If NRSQ = 0, the centrifugal operator is diagonal:

- If IBOUND = 0, its diagonal matrix elements are calculated from the L array as $\hbar^2$L(L + 1)/(2μR^2). In this case, CPL9 should leave the array CENT unchanged.
- If IBOUND $\neq$ 0, CPL9 should return an array of values in the array CENT such that the diagonal elements of the centrifugal operator are $\hbar^2$ CENT/(2μR^2).

For diagonal basis sets, NCONST = 0. CPL9 can return NDGVL blocks of diagonal contributions to the pair energy, each of dimension N, in the array DGVL. Each block is then multiplied by the matching member of the VCONST array, giving NDGVL contributions to the pair energy. DGVL may therefore be used to include EFV-dependent contributions to the pair energy as described in section 17.4.

17.10 Additional subroutines required in some cases

17.10.1 DEGEN9: denominators for degeneracy-averaged cross sections

Routine DEGEN9 is required only when calculating degeneracy-averaged state-to-state cross sections in MOLSCAT. Its specification is:

```
SUBROUTINE DEGEN9(JJ1, JJ2, DEGFAC)

DOUBLE PRECISION, INTENT(OUT) :: DEGFAC

INTEGER,          INTENT(IN)  :: JJ1, JJ2
```

JJ1 and JJ2 are the pair level indices of the initial and final levels. The routine must return the degeneracy factor g_{n_i}, used in the numerator of Eq. 2.12 for degeneracy-averaged cross sections, in argument DEGFAC.

For NCONST > 0, levels for cross-section calculations are identified by comparing threshold energies. All cases of this implemented so far use the loop over JTOT for $M_{\rm tot}$ and set JHALF = 0. In this case it is usually sufficient to use a routine that sets DEGFAC to 1.0 regardless of the values of JJ1 and JJ2 (this is supplied in degen9-nondegenerate.f). However, this does not take account of identical-particle symmetry and may also give incorrect results if DEGTOL is too large to distinguish all levels uniquely.

17.10.2 THRSH9: threshold energies from monomer quantum numbers

If NCONST > 0 and the user wishes to specify reference energies from monomer quantum numbers input in the array MONQN, the routine THRSH9 must be provided. If reference energies

are instead specified using either EREF or positive values for IREF, the dummy version of THRSH9, which is supplied in thrsh9-dummy.f, is sufficient.

The specification of THRSH9 is:

```
SUBROUTINE THRSH9(IREF, MONQN, NQN, EREF, IPRINT)

DOUBLE PRECISION, INTENT(OUT) :: EREF

INTEGER,          INTENT(IN)  :: IREF, MONQN(NQN), NQN, IPRINT
```

This subroutine is called by THRESH if NCONST > 0 and MONQN is specified in namelist &INPUT. It must calculate the energy of the threshold identified by the quantum numbers in MONQN and place the resulting value in EREF (in units of cm^{-1}, irrespective of EUNITS or EUNIT).

IPRINT may be used to control the level of output from THRSH9, with higher values producing increased amounts of output. If THRSH9 needs access to the current values of external fields, it should obtain them from the module efvs as described in section 17.5.

17.10.3 EFV9: Converting EFVs to values in the VCONST array

External fields are handled as part of H_{intl}, which is expanded as

$$H_{\mathrm{intl}}(\xi_{\mathrm{intl}}) = \sum_{\Omega} h_{\Omega} \mathcal{H}^{\Omega}_{\mathrm{intl}}(\xi_{\mathrm{intl}}). \tag{17.8}$$

The R-independent coefficients h_{Ω} are held as elements of the array VCONST described in 17.5.

In the simplest cases, there is a one-to-one correspondence between the EFVs and (a sequential subset of) values in VCONST. In this case, all that is required is to set MAPEFV (returned from BAS9IN) to the index of the first EFV in the VCONST array. Subsequent EFVs simply correspond to subsequent elements of VCONST, so that

$$\mathtt{VCONST}(i + \mathtt{MAPEFV} - 1) = \mathtt{EFV}(i) \qquad \text{for all } i \in [1, \mathtt{NEFV}].$$

In the trivial case of a single EFV, MAPEFV is simply its index in the VCONST array.

In more complicated cases, the programmer may wish to implement non-linear relationships between the input EFVs and the coefficients h_{Ω} in the array VCONST. For example, one of the EFVs might be an angle between a field and the quantisation axis. In this case routine EFV9 may be provided to specify the relationship.

The specification of EFV9 is:

```
SUBROUTINE EFV9(IFVARY)
USE efvs
USE potential

INTEGER, INTENT(IN) :: IFVARY
```

EFV9 is called by SETEFV if MAPEFV < 0 or IFVARY < 0. It must set values for the relevant elements of the array VCONST (in module potential) from the values of EFVs stored in the EFV array in module efvs. The value of MAPEFV is also included in module efvs.

EFV9 should perform two sequential operations:

1. If IFVARY is negative, the single EFV being varied (to characterise resonances in MOLSCAT or locate bound states in FIELD) is a proxy EFV that affects more than one element of the EFV array. It is stored in EFV(NEFV+1). EFV9 must use this value to set the values of EFV(1:NEFV) as required. If several different mappings are required, each one can be implemented for a different negative value of IFVARY.
2. If MAPEFV is negative, the EFVs are not in a one-to-one correspondence with the coupling coefficients. EFV9 must use the values of EFV(1:NEFV) to set values in the VCONST array as required.

17.11 Resolving threshold degeneracies with extra operators

The requirement that the asymptotic basis functions are eigenfunctions of $H_{\rm intl}$ and $\hat{L}^2$ is not always enough to define them uniquely, because of degeneracies or near degeneracies. Under these circumstances, a basis-set suite may construct coupling matrices for extra operators to be used in resolving the degeneracies, as described in section 9.2.

To use this facility, the programmer must set the variable NEXTRA and the array NEXTMS (in module potential) in BAS9IN; the values required are described in section 18.2.2. The extra terms must come *after* any terms that contribute to the Hamiltonian. CPL9 must calculate coupling matrices for each of the extra terms. The coupling coefficients stored in VCONST for any terms that do not contribute to the Hamiltonian are ignored.

17.12 Modules available for use in plug-in basis-set suites

17.12.1 Module basis_data

The specification of module basis_data is

```
USE sizes, ONLY: MXELVL, MXJLVL, MXROTS, MXSYMS

INTEGER          :: IDENT, JHALF, ISYM(MXSYMS), ISYM2(MXSYMS), JMIN, J2MIN, &
                    JMAX, J2MAX, JSTEP, J2STEP, JLEVEL(MXJLVL), NJLQN9,     &
                    NLEVEL

DOUBLE PRECISION :: ELEVEL(MXELVL), EMAX, ROTI(MXROTS), SPNUC, WT(2)
```

The array dimensions MXROTS, MXSYMS, MXELVL and MXJLVL are set in module sizes, and are currently 12, 10, 1000 and 4000 respectively.

17.12.2 Module `potential`

The specification of module `potential` is

```
USE sizes, ONLY: MXOMEG, MXLMDA

INTEGER          :: NDGVL, NCONST, NRSQ, IREF, NVLBLK, NEXTRA, NEXBLK,     &
                    NEXTMS(MXOMEG), LAMBDA(MXLMDA)

DOUBLE PRECISION :: VCONST(MXOMEG)

CHARACTER(10)    :: RMNAME, EPNAME
```

The array dimensions `MXOMEG` and `MXLMDA` are set in module `sizes` and are currently 20 and 2000 respectively.

The `VL` array consists of `NVLBLK` coupling matrices. The first `MXLAM` of these are coupling matrices for the R-dependent terms in the potential. For non-diagonal H_{intl}, these are followed by `NCONST` coupling matrices for the R-independent terms of the internal Hamiltonian plus any terms needed for interactions with external fields, and then `NRSQ` coupling matrices for the centrifugal term. The remaining `NEXBLK` (if any) are used for extra operators.

For diagonal H_{intl}, there may be `NDGVL` diagonal terms that contribute to the interaction energy as described in section 17.4.

The array `LAMBDA` is included in this module so that its contents are available for constructing the `VL` array if required.

The character variables `RMNAME` and `EPNAME` are included in this module to allow plug-in potential routines to return names for the length and energy units they use, for printing elsewhere in the programs.

17.12.3 Module `efvs`

The specification of module `efvs` is

```
INTEGER, PARAMETER :: MXEFV=10, LEFVN=20, LEFVU=6

INTEGER            :: NEFV, ISVEFV, MAPEFV, LISTFV(1:MXEFV+1), NNZRO,       &
                      IEFVST, NEFVP

DOUBLE PRECISION   :: EFV(0:MXEFV)

CHARACTER(LEFVN)   :: EFVNAM(0:MXEFV), SVNAME
CHARACTER(LEFVU)   :: EFVUNT(0:MXEFV), SVUNIT

DOUBLE PRECISION   :: SCALAM
```

```
INTEGER             :: ITPSUB
```

At any point in the programs, the array `EFV` contains the current values of all the external fields, with `EFV(0)` containing the current value of the potential scaling factor. `EFVNAM(0)` and `EFVNAM(NEFV+1)` are set internally to be `POTL SCALING FACTOR` and `PROXY VARIABLE` respectively.

The array `LISTFV` (of dimension `NNZRO`) and the variables `IEFVST` and `NEFVP` are used internally for printing the EFVs. `IEFVST` is set internally to be $\min\{0, \texttt{ISVEFV}\}$ and `NEFVP` is set internally to be `NEFV` unless there is a proxy EFV, in which case it is `NEFV+1`.

`SVNAME` and `SVUNIT` are set internally to be `EFVNAM(ISVEFV)` and `EFVUNT(ISVEFV)` respectively.

The potential scaling factor `SCALAM` is also declared in this module, as it is handled in the same way as the EFVs.

The variable `ITPSUB` may be set by a plug-in basis-set suite and is output on unit `ISAVEU` so that an external program can identify the basis-set suite that produced the results.

Chapter 18

Supplied plug-in basis-set suites

We have provided two example basis-set suites, which are described in this chapter.

18.1 ^1S atom + $^3\Sigma$ diatom

The basis-set suite provided in file `base9-1S_3Sigma_cpld.f` handles an atom in a ^{1}S state interacting with a diatomic molecule in a $^3\Sigma$ state, in the presence of an external magnetic field. The internal Hamiltonian $H_{\rm intl}$ is composed of 4 parts, all for the diatomic molecule [50],

$$H_{\rm intl} = H_{\rm rot} + H_{\text{spin-rot}} + H_{\text{spin-spin}} + H_{\rm Z}, \tag{18.1}$$

where

$$H_{\rm rot} = B_v \hat{n}^2; \tag{18.2}$$

$$H_{\text{spin-rot}} = \gamma \hat{\boldsymbol{s}} \cdot \hat{\boldsymbol{n}}; \tag{18.3}$$

$$H_{\text{spin-spin}} = \frac{2}{3}\lambda\sqrt{\frac{24\pi}{5}}\sum_q (-1)^q Y_2^{-q}(r)[\hat{\boldsymbol{s}} \otimes \hat{\boldsymbol{s}}]^2_q; \text{ and} \tag{18.4}$$

$$H_{\rm Z} = -\hat{\boldsymbol{\mu}} \cdot \mathbf{B} \quad (\text{where } \boldsymbol{\mu} = -g_S \mu_{\rm B} \hat{\boldsymbol{s}}). \tag{18.5}$$

We use lower-case letters for the angular momentum operators to indicate that they operate on only one of the two species involved (the diatom in this case, since the atom is structureless). The quantity referred to as g_e in ref. [50] is positive, so is denoted g_S here.

The values of B_v, γ and λ are input in `ROTI`(1–3), in the units specified by `EUNITS` or `EUNIT`. The external magnetic field is controlled by `FLDMIN` and `FLDMAX`, which are taken to be input in units of G.

The basis set implemented in this suite is $|(n,s)j,m_j\rangle|L,M_L\rangle$. Here n is the rotational quantum number for the diatomic molecule and s is its spin. These are coupled to form a resultant j, with projection m_j onto the Z axis defined by the magnetic field. However, j is *not* coupled to the end-over-end angular momentum of the pair L to form a total angular momentum $J_{\rm tot}$. $J_{\rm tot}$ is not a good quantum number in the presence of a magnetic field. $M_{\rm tot} = m_j + M_L$, however, is a good quantum number and there is a separate set of coupled equations for each value of $M_{\rm tot}$. The loop over `JTOT` is used for $M_{\rm tot}$, which runs from `JTOTL` to `JTOTU`.

The spin, s, is the same for all basis functions and M_L is defined by $M_{\rm tot}$ and m_j, so each basis function (for a given $M_{\rm tot}$) is specified by values of n, j and m_j and L. L is held in the separate array `L`, so the three quantum numbers that label each pair state are n, j and m_j.

The internal Hamiltonian is nearly diagonal in this basis set. The only off-diagonal terms are due to $H_{\rm spin\text{-}spin}$, and have the selection rule $\Delta n = \pm 2$. These terms are important for Feshbach resonances, but have only a small effect on energy levels. The basis-set suite has an option to neglect them, controlled by `IBSFLG`.

Routine `BAS9IN` is called first. It sets default values for its input variables and then reads namelist (`&BASIS9`), which contains the quantities

`IS` is the spin s

`LMAX` is the maximum value for L in the basis set

`LMIN` is the minimum value for L in the basis set

`IBSFLG` controls whether off-diagonal terms in the monomer Hamiltonian are to be included: 2=yes, 1=no; default is yes

`MLREQ` can be used to restrict the basis set to functions with a single required value of M_L (default is to include all)

18.1.1 Additional information for programmers

The remainder of this subsection is mostly for programmers who wish to understand this basis-set suite as an aid to programming their own.

`BAS9IN` sets the label (`'ATOM + 3SIGMA IN MAGNETIC FIELD'`) for the interaction type. If `IBSFLG` = 1, it sets `NCONST` to 0, `MAPEFV` to 1 and `NDGVL` to 1. In this case, the diagonal part of $H_{\rm Z}$ is contained in the `DGVL` array. If `IBSFLG` = 2, `BAS9IN` sets `NCONST` = 4, indicating that $H_{\rm intl}$ is described using 4 blocks of the array `VL` as described under `CPL9` below. In this case it also sets `MAPEFV` = 4, indicating that the coupling matrix for the Zeeman interaction with the external magnetic field is held in the 4th block of `VL`. `NQN` is always set to 4 and `NEFV` is always set to 1.

Routines `SET9`, `BASE9`, `CPL9`, `THRSH9` and `DEGEN9` are coded as entry points in subroutine `BAS9IN` so that that they have access to the same list of internal quantities. They also have access to quantities read in namelist `&BASE` via module `basis_data`.

`SET9` loops over values of n from `JMIN` to `JMAX` in steps of `JSTEP`, over j from $|n-s|$ to $n+s$, and over m_j from $-j$ to j. It places values of n, j and m_j in `JLEVEL` and then

populates the ELEVEL array with the field-free diagonal elements of the internal Hamiltonian $H_{\rm intl} = H_{\rm rot} + H_{\rm spin\text{-}rot} + H_{\rm spin\text{-}spin}$. JLEVEL and ELEVEL are used outside the basis-set suite only if NCONST = 0 (i.e., for IBSFLG = 1 but not for IBSFLG = 2). However, SET9 copies the elements of JLEVEL into the array JSTATE, which is used externally for either value of IBSFLG; note that JSTATE is structured differently from JLEVEL.

BASE9 sets up the JSINDX and L arrays for the current combination of $M_{\rm tot}$ and symmetry block IBLOCK (which is used for the total parity in this suite). For each pair state, L runs from LMIN to LMAX, but only functions of the required total parity $(-1)^{n+L}$ are included: parity -1 for IBLOCK = 1 and parity $+1$ for IBLOCK = 2. For each basis function, m_j implies a value of $M_L = M_{\rm tot} - m_j$; only values $L \le |M_L|$ are included.

CPL9 sets up the VL array and (if IBSFLG = 1) the DGVL array. The first MXLAM blocks of the VL array contain the coupling matrices for the Legendre polynomials used in the expansion of the interaction potential; see equation 13 of [50] for the explicit expression.

If IBSFLG = 2, the next 4 blocks contain the coupling matrices for $H_{\rm rot}$, $H_{\rm spin\text{-}rot}$, $H_{\rm spin\text{-}spin}$ and $H_{\rm Z}$, respectively; together these make up $H_{\rm intl}$. They are defined with corresponding prefactors h_Ω (as in Eq. 17.8) B_v, $\gamma\sqrt{s(s+1)(2s+1)}$, $\lambda\frac{2}{3}\sqrt{30}$ (in cm^{-1}) and the magnetic field (in Gauss), respectively; these prefactors are held in the array VCONST and all other factors are absorbed into the operators $\boldsymbol{\mathcal{H}}^\Omega$ whose matrix elements are in the array VL.

If IBSFLG = 1, there are only MXLAM blocks of the VL array, but the DGVL array contains the diagonal part of $H_{\rm Z}$; the corresponding prefactor, held in VCONST, is the magnetic field (in Gauss). The pair energies for a particular magnetic field are calculated in CHEINT.

THRSH9 calculates the energy of the $^3\Sigma$ molecule in a magnetic field from quantum numbers (n, j, m_j) supplied in the array MONQN. If IBSFLG = 1, it neglects off-diagonal matrix elements of $H_{\rm spin\text{-}spin}$. If IBSFLG = 2, it constructs and diagonalises a 2×2 monomer Hamiltonian matrix if necessary (i.e., if $j = n \pm 1$), taking account of the basis set size specified by JMAX.

18.2 Alkali-metal atom + alkali-metal atom

The basis-set suite provided in file base9-alk_alk_ucpld.f handles interactions between two alkali-metal atoms in ^{2}S states in a magnetic field, including hyperfine interactions.

The internal Hamiltonian is composed of two parts,

$$H_{\rm intl} = H_{\rm hyperfine} + H_{\rm Z}, \tag{18.6}$$

where

$$H_{\rm hyperfine} = h \sum_{x=A,B} \zeta_x \hat{\boldsymbol{\imath}}_x \cdot \hat{\boldsymbol{s}}_x; \qquad H_{\rm Z} = \mu_{\rm B} \sum_{x=A,B} (g_{S_x} \hat{\boldsymbol{s}}_x + g_{n_x} \hat{\boldsymbol{\imath}}_x) \cdot \mathbf{B}. \tag{18.7}$$

We use lower-case letters for the angular momentum operators to indicate that each one operates on the spins of just one atom.

The hyperfine coupling constants ζ_x are input as hyperfine splittings $\Delta W_x = \zeta_x(i_x + 1/2)$, in frequency units (GHz). The external magnetic field is controlled by FLDMIN and FLDMAX, which are taken to be input in units of G.

The basis set implemented in this suite uses an uncoupled basis set for each atom, $|\gamma\rangle = |s, m_s\rangle|i, m_i\rangle$, where s is the electronic spin, i is the nuclear spin, and m_s and m_i are the corresponding projections onto the Z axis defined by the magnetic field. The basis set for the pair is $|\gamma_A\rangle|\gamma_B\rangle|L, M_L\rangle$, where L is the end-over-end angular momentum of the pair and M_L is its projection. The total angular momentum $J_{\rm tot}$ is not a good quantum number in the presence of a magnetic field. Instead, $M_{\rm tot} = m_{sA} + m_{iA} + m_{sB} + m_{iB} + M_L$ is a good quantum number and there is a separate set of coupled equations for each value of $M_{\rm tot}$. Since $M_{\rm tot}$ can be half-integer, the loop over `JTOT` is used for the *doubled* quantum number $2M_{\rm tot}$, which runs from `JTOTL` to `JTOTU` in steps of `JSTEP`.

For each basis function, M_L is defined by $M_{\rm tot}$, m_{sA}, m_{iA}, m_{sB} and m_{iB}. The values of s_A, i_A, s_B and i_B are the same for all basis functions, so each basis function (for a given $M_{\rm tot}$) is specified by values of m_{sA}, m_{iA}, m_{sB}, m_{iB} and L. L is held in a separate array, so the four quantum numbers that label each pair state are m_{sA}, m_{iA}, m_{sB} and m_{iB}.

Routine `BAS9IN` is called first. It sets default values for its input variables and then reads namelist (`&BASIS9`), which contains the quantities

`ISA` is (double) the electronic spin of atom A;

`ISB` is (double) the electronic spin of atom B;

`INUCA` is (double) the nuclear spin of atom A;

`INUCB` is (double) the nuclear spin of atom B. If set to a negative value, atoms A and B are taken to be identical;

`GSA` is the electronic g-factor for atom A in bohr magnetons;

`GSB` is the electronic g-factor for atom B in bohr magnetons;

Note that the last two quantities are positive: they correspond to values for g_S, not g_e. For high-precision work on alkali-metal atoms it is usual to use high-precision values for the unpaired electron of the specific isotope, as tabulated (for example) by Steck.

`GA` is the nuclear g-factor for atom A in bohr magnetons;

`GB` is the nuclear g-factor for atom B in bohr magnetons;

These two quantities are defined with the same sign convention as g_S, following Arimondo [82].

`HFSPLA` is the hyperfine splitting for atom A in GHz;

`HFSPLB` is the hyperfine splitting for atom B in GHz;

`LMAX` is the maximum value for L to be included in the basis set;

`NREQ` The basis may be limited to functions with values of L and M_F in a specified list. If `NREQ` > 0 (maximum 10) values of L and M_F are used to constrain the basis functions:

`LREQ` is a list of values of L to be included in the basis;

`MFREQ` is a (matching) list of values of M_F to be included in the basis; a value of -999 includes all M_F values for this L;

`ISPSP` If non-zero then the spin-spin term is included in the coupled equations.

18.2.1 Additional information for programmers

The remainder of this subsection is mostly for programmers who wish to understand this basis-set suite as an aid to programming their own.

`BAS9IN` sets the label (`'ATOM - ATOM WITH NUCL SP + MAG FL'`) for the interaction type. It sets `NCONST` $= 2$, indicating that $H_{\rm intl}$ is described using 2 blocks of the array `VL` as described under `CPL9` below. It also sets `MAPEFV` $= 2$, indicating that the coupling matrix for the Zeeman interaction with the external magnetic field is held in the 2nd of these blocks. `NQN` is set to 5.

Routines `SET9`, `BASE9`, `CPL9` and `DEGEN9` are coded as entry points in subroutine `BAS9IN` so that that they have access to the same list of internal quantities. They also have access to quantities read in `&BASE` in module `basis_data`.

`SET9` loops over all possible values of m_{sA}, m_{iA}, m_{sB}, m_{iB} for the supplied values of s_A, i_A, s_B and i_B. It places doubled values of m_{sA}, m_{iA}, m_{sB}, m_{iB} into `JSTATE`.

`BASE9` sets up the arrays `JSINDX` and `L` for the current combination of $M_{\rm tot}$ and symmetry block `IBLOCK` (which is used for total parity in this suite). For each pair state, `L` runs up to `LMAX`, but only functions of the required total parity $(-1)^{n+L}$ are included: parity $= -1$ for `IBLOCK` $= 1$ and parity $= +1$ for `IBLOCK` $= 2$. For each basis function, the projections m_{sA}, m_{iA}, m_{sB}, m_{iB} imply a value of $M_L = M_{\rm tot} - m_j$; only values $L \le |M_L|$ are included. If `NREQ` is set, basis functions are also excluded if they do not satisfy the values in the arrays `LREQ` and `MFREQ`.

If the two alkali-metal atoms are identical, the basis set is symmetrised with respect to atom exchange. The basis functions are then

$$\frac{1 \pm (-1)^L P_{AB}}{\left[2\left(1+\delta_{m_{sA}m_{sB}}\delta_{m_{iA}m_{iB}}\right)\right]^{1/2}} \left|m_{sA}m_{iA}m_{sB}m_{iB}LM_L\right\rangle \tag{18.8}$$

with the $+$ sign for bosons and the $-$ sign for fermions. The operator P_{AB} exchanges all A-labelled functions with their B-labelled counterparts. Routine `BASE9` simply avoids duplicating pair functions related by exchange symmetry and excludes symmetry-forbidden pair functions; the actual symmetrisation is handled in `CPL9`.

`CPL9` sets up the `VL` array using the formulas given in the appendix of [83]. The first `NSPIN` $= 1 + 2\min\{s_A, s_B\}$ blocks contain the coupling matrices for the interaction potentials (singlet and triplet for $s_A = s_B = \frac{1}{2}$). The next block contains the coupling matrix for the spin-spin (dipolar) term, which is R-dependent so is handled as a potential term. The scaling of the spin-spin term is described below. The next `NCONST` $= 2$ blocks contain the coupling matrices for the the atomic hyperfine term and Zeeman term, respectively; together these make up $H_{\rm intl}$. They are defined with corresponding prefactors h_Ω (as in Eq. 17.8) 1/29.99792458 (conversion

from GHz to cm^{-1}) and the magnetic field (in Gauss), respectively; these prefactors are held in the array VCONST and all other factors (including hyperfine coupling constants and g-factors) are absorbed into the operators $\boldsymbol{\mathcal{H}}^{\Omega}$ whose matrix elements are in the array VL.

The spin-spin dipolar term is

$$\hat{V}^{\mathrm{d}}(R) = \lambda(R)\left[\hat{s}_1 \cdot \hat{s}_2 - 3(\hat{s}_1 \cdot \vec{e}_R)(\hat{s}_2 \cdot \vec{e}_R)\right], \tag{18.9}$$

where $\vec{e}_R$ is a unit vector along the internuclear axis. The coefficient $\lambda(R)$ is commonly written in a form such as

$$\lambda(R) = -E_{\mathrm{h}}\alpha^2\left[\frac{-g_S^2}{4(R/a_0)^3} - A\exp(-\beta R/a_0)\right]. \tag{18.10}$$

The factor $-E_{\mathrm{h}}\alpha^2$ is included in the VL array. The code was originally designed for use with potential routines that return the singlet and triplet potentials as wavenumbers in cm^{-1}; in this case the spin-spin potential coefficient returned by POTENL should be the dimensionless quantity in brackets in Eq. 18.10. However, if POTENL works in different units (EPSIL $\neq 1.0$), this quantity must be divided by EPSIL.

The matrix elements of the electronic singlet and triplet potentials are calculated by the function CENTPT. Then, if ISPSP is non-zero, the spin-spin term is calculated by the function SPINSP. The matrix elements for the hyperfine interaction are calculated by the function SDOTI2. The Zeeman interaction is diagonal in this basis set and very simple, so is calculated in-line.

Subroutine POTIN9 sets ITYPE to 1. MXLAM must be set to be $1 + 2\min\{s_A, s_B\}$ in namelist &POTL.

DEGEN9 sets the degeneracy factor for all levels to be 1.

THRSH9 calculates the threshold energy for an atom pair with specified quantum numbers from a separate calculation of the energies of the two atoms in a magnetic field. The present implementation assumes that the electronic spin of each atom is 1/2. It calculates the threshold corresponding to the 2 atomic hyperfine states indicated by the values in the MONQN array which correspond to $2f_A$, $2m_{fA}$, $2f_B$, $2m_{fB}$; f_A and f_B are not good quantum numbers for the atomic states at finite magnetic field, but are interpreted to mean the upper and lower states that *correlate* with f_A and f_B at zero field.

18.2.2 Extra operator functionality

The basis-set suite supplied for alkali-alkali interactions includes two extra operators for resolving degeneracies, as an illustration of how to program and use such operators.

The general format for extra operators is specified by the variable NEXTRA and the array NEXTMS, with NEXTRA elements. The IEXTRAth extra operator has NEXTMS(IEXTRA) terms.

The two extra operators in the supplied CPL9 are designed to be diagonal in the eigenbasis, with different eigenvalues for energetically degenerate thresholds. They are diagonal in the uncoupled basis set, with diagonal matrix elements $m_{fA}^2 + m_{fB}^2$ and $m_{fA} + m_{fB}$ respectively (where $m_{fA} = m_{sA} + m_{iA}$ etc). POTIN9 sets NEXTMS = 1, 1.

In the supplied basis-set suite, namelist `&BASIS9` includes an additional item `NEXTRA` to control which extra operators are used. Valid values are 0 (no extra operators), 1 (just the first extra operator above) and 2 (both extra operators above). `NEXTRA` defaults to 0.

Chapter 19

Supplied potential routines

This chapter documents potential routines supplied with the programs that may be generally useful. It does not include the routines supplied for the rare gas - CH_4 and H_2 - H_2 systems, which are for demonstration purposes only.

19.1 Potential energy surfaces for rare gas - H_2 systems

The special-purpose `POTENL` routine in file `potenl-Rg_H2.f` evaluates the BC3(6,8) potentials of Le Roy and Carley [72] and the TT3(6,8) potential of Le Roy and Hutson [81] for rare gas + H_2 systems.

Input data for the BC3(6,8) potential for Ar-H_2 are supplied in the file `molscat-Ar_H2.input`, and for the TT3(6,8) potential in `bound-Ar_H2.input`.

The routine uses tabulations of monomer matrix elements between H_2 rovibrational states that are read from the file `data/h2even.dat`. Data files `h2odd.dat`, `d2even.dat` and `hd.dat` are available on request.

19.2 Potential energy surfaces for Ar-HF and Ar-HCl

The routines in file `extpot-Rg_HX.f` evaluate the H6(4,3,2) potential of Hutson for Ar-HF [69] and H6(4,3,0) potential of Ar-HCl [79] at fixed values of R and θ. They are called via a version of `VRTP` in file `vrtp-Rg_HX-eta.f` that reads supplementary potential data from the main input file. These routines can also evaluate older potentials for the rare gas + HX systems [84, 85, 86, 87]. The general-purpose version of `POTENL` performs numerical quadrature over θ to evaluate the coefficients $V_\lambda(R)$ of the expansion in Legendre polynomials.

Input data for the H6(4,3,2) potential for Ar-HF are supplied in the file `molscat-Ar_HF.input`, and for the H6(4,3,0) potential for Ar-HCl in `bound-Ar_HCl.input`.

The H6 potentials depend parametrically on the *mass-reduced vibrational quantum number* $(v+\frac{1}{2})/\sqrt{\mu}$, where v and μ are the vibrational quantum number and reduced mass of the state

of HF or HCl (or DF or DCl) required. To produce results for different vibrational states, the diatom rotational constant must be changed appropriately in ROTI and the centre-of-mass shift, bond length, partial charges and mass-reduced quantum number must be changed in the last 2 lines of each potential input data. The values required for HF are given in table II of ref. [69], and those required for HCl are given in table II of ref. [79].

19.3 Potential energy surfaces for Ar-CO_2

The routines in file `extpot-Ar_CO2.f` evaluate the single-repulsion and split-repulsion potentials of Hutson *et al.* [70] for Ar-CO_2. They are called via a version of `VRTP` in file `vrtp-extpot_1ang.f`. The general-purpose version of `POTENL` performs numerical quadrature over θ to evaluate the coefficients $V_\lambda(R)$ of the expansion in Legendre polynomials.

Input data for the single-repulsion potential are supplied in the file `molscat-Ar_CO2.input`, and for the split-repulsion potential in `bound-Ar_CO2.input`.

19.4 Potential energy surface for Mg-NH

The routines in file `vstar-Mg_NH.f` supply the interaction potential of Soldán *et al.* [76] for Mg+NH. They read potential points evaluated at fixed values of R and θ, at Gauss-Lobatto quadrature points in θ. They project out the coefficients of the Legendre expansion by Gauss-Lobatto quadrature, and interpolate the resulting coefficients by RKHS interpolation [88, 89], imposing analytical power-series representations on coefficients at long range [89].

19.5 Potential curves in the form of Tiemann and coworkers

The routines in file `vstar-Tiemann.f` implement the functional forms that have been used by Tiemann and coworkers to fit interaction potentials for a wide range of diatomic molecules, including several alkali-metal diatomic molecules. They are implemented as `VINIT`, `VSTAR` and `VSTAR1` routines that take potential parameters in the form of a module named `pot-data-tiemann`. The data module provided in `pot_data_Tiemann-Rb2-2010.f` includes data for the Rb_2 potentials of Strauss *et al.* [78], but the routines can also be used (with different versions of the `pot-data-tiemann` data module) for many other systems. They are used in the MOLSCAT and FIELD executables for the example input files `molscat-basic_Rb2.input`, `molscat-Rb2.input`, `field-basic_Rb2.input` and `field-Rb2.input`.

The `POTENL` routine calls the following internal subunits:

subroutine	found in
VINIT	vstar-Tiemann.f
VSTAR	vstar-Tiemann.f
VSTAR1	vstar-Tiemann.f

subroutine	found in
VSTAR2	vstar-Tiemann.f
POWER	vstar-Tiemann.f
DPOWER	vstar-Tiemann.f

The functional form in the well region of the potential, between a short-range limit $r_{S,\mathrm{sr}}$ and

a long-range limit $r_{S,\mathrm{lr}}$, is a series expansion

$$\sum_{i=0}^{n_{S,\mathrm{exp}}} a_{S,i}[\xi_S(r)]^i, \text{ where } \xi_S(r) = \frac{r - r_{S,\mathrm{m}}}{r + b_S r_{S,\mathrm{m}}} \tag{19.1}$$

with different sets of parameters for each total spin $S = 0$ or 1. $r_{S,\mathrm{m}}$ is chosen to be near the equilibrium distance of the state with multiplicity $2S + 1$. The potential is extrapolated to short range ($r < r_{S,\mathrm{sr}}$) with the form $A_S + B_S(r/a_0)^{-n_{S,\mathrm{sr}}}$, where a_0 is the Bohr radius, and to long range ($r > r_{S,\mathrm{lr}}$) with $\sum_{i=6}^{n_{\mathrm{lr}}} -C_i/r^i + (-1)^{S+1}V_{\mathrm{exch}}(r)$. The dispersion coefficients, C_i, are common to both curves and the long-range exchange contribution is $V_{\mathrm{exch}}(r) = A_{\mathrm{ex}}(r/a_0)^\gamma \exp(-\beta r/a_0)$.

There are a number of constraints that are often applied to the parameters, but these have sometimes been relaxed for specific published potential curves. These constraints are therefore optional, and controlled by values in the data module:

`GAMBET` The theoretical form of the long-range exchange function [90] suggests that β and γ are related by $\gamma = 7/\beta - 1$. If `GAMBET` = 1, γ is calculated from the input β; if `GAMBET` = 2, β is calculated from the input γ; otherwise, the supplied values for β and γ are used unchanged.

`MATCHL` $a_{S,0}$ is chosen to make the mid-range polynomial match the value of the long-range potential at $r_{S,\mathrm{lr}}$. However, published values of $a_{S,0}$ are inevitably rounded. If `MATCHL` is `.TRUE.`, $a_{S,0}$ is recalculated to make the potential curve exactly continuous at $r_{S,\mathrm{lr}}$. If `MATCHL` is `.FALSE.`, $a_{S,0}$ is left at the value in the data module and the extent of the resulting mismatch is reported in the printed output. There is by construction a discontinuity in the potential *derivative* at $r_{S,\mathrm{lr}}$, and the extent of this is also reported.

`MATCHD` B_S is often chosen to make the radial derivative of the short-range potential match that of the mid-range polynomial at $r_{S,\mathrm{sr}}$. If `MATCHD` is `.TRUE.`, B_S is recalculated to make the potential derivative exactly continuous at $r_{S,\mathrm{sr}}$. If `MATCHD` is `.FALSE.`, B_S is left at the value in the data module and the extent of the resulting mismatch is reported in the printed output.

`MATCHV` A_S is chosen to make the short-range potential match the value of the mid-range polynomial at $r_{S,\mathrm{sr}}$. However, published values of A_S are inevitably rounded. If `MATCHV` is `.TRUE.`, A_S is recalculated to make the potential curve exactly continuous at $r_{S,\mathrm{sr}}$. If `MATCHV` is `.FALSE.`, A_S is left at the value in the data module and the extent of the resulting mismatch is reported in the printed output.

`NEX` **and** `CEX` Some published potentials have an extra term in the "long-range" expansion of the form $-C_N/r^N$. If this is present, the power and coefficient are set in `NEX` and `CEX`.

The appropriate data module must be linked with the subroutine `VINIT`, which has entry points `VSTAR` and `VSTAR1`. `VSTAR` and `VSTAR1` evaluate the value and derivative of either the singlet or the triplet potential at `R`. The functions `POWER` and `DPOWER` are used to evaluate a power series and its derivative, respectively.

Acknowledgements

We are grateful to an enormous number of people who have contributed routines, ideas, and comments over the years. Any attempt to list them is bound to be incomplete. Many of the early contributors are mentioned in the program history in section 1.8. In particular, we owe an enormous debt to the late Sheldon Green, who developed the original MOLSCAT program and established many structures that have proved general enough to support the numerous later developments. He also developed the `DCS` and `SBE` post-processors. Robert Johnson, David Manolopoulos, Millard Alexander, Gregory Parker and George McBane all contributed propagation methods and routines. Christopher Ashton added code to calculate eigenphase sums and developed the `RESFIT` post-processor. Timothy Phillips developed code for interactions between asymmetric tops and linear molecules. Alice Thornley developed methods to calculate bound-state wavefunctions from log-derivative propagators and George McBane extended them to scattering wavefunctions. Maykel Leonardo González-Martínez worked on the addition of structures for non-diagonal Hamiltonians, including magnetic fields, and Matthew Frye contributed algorithms for converging on quasibound states as a function of energy and on low-energy Feshbach resonances (both elastic and inelastic) as a function of external field.

Development of version 2020.0 of the programs was supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC) under Grant Nos. EP/P01058X/1, EP/P008275/1 and EP/N007085/1.

Chapter 20

Bibliography

[1] S. Green. 'MOLSCAT molecular scattering program, version 7.' NRCC Software Catalog **1**, KQ01 (1980).

[2] S. Green and J. M. Hutson. 'DCS computer program, version 2.0.' Distributed by Collaborative Computational Project No. 6 of the UK Engineering and Physical Sciences Research Council (1996).

[3] J. M. Hutson and S. Green. 'SBE computer program.' Distributed by Collaborative Computational Project No. 6 of the UK Engineering and Physical Sciences Research Council (1982).

[4] J. M. Hutson. 'RESFIT 2007 computer program.' (2007).

[5] J. M. Hutson and S. Green. 'MOLSCAT computer program, version 14.' Distributed by Collaborative Computational Project No. 6 of the UK Engineering and Physical Sciences Research Council (1994).

[6] G. C. McBane. 'PMP Molscat.' (2005). [Online; accessed 1-December-2017, last updated 12-October-2011].

[7] J. M. Hutson. 'Coupled-channel methods for solving the bound-state Schrödinger equation.' Comput. Phys. Commun. **84**, 1 (1994).

[8] J. M. Hutson. 'BOUND computer program, version 5.' Distributed by Collaborative Computational Project No. 6 of the UK Engineering and Physical Sciences Research Council (1993).

[9] D. E. Manolopoulos and S. K. Gray. 'Symplectic integrators for the multichannel Schrödinger equation.' J. Chem. Phys. **102**, 9214 (1995).

[10] M. P. Calvo and J. M. Sanz-Serna. 'The development of variable-step symplectic integrators, with application to the two-body problem.' SIAM J. Sci. Comput. **14**, 936 (1993).

[11] R. I. McLachlan and P. Atela. 'The accuracy of symplectic integrators.' Nonlinearity **5**, 541 (1992).

[12] M. D. Frye and J. M. Hutson. 'Characterizing quasibound states and scattering resonances.' arXiv:1912.00206 (2019).

[13] M. D. Frye and J. M. Hutson. 'Characterizing Feshbach resonances in ultracold scattering calculations.' Phys. Rev. A **96**, 042705 (2017).

[14] J. M. Hutson. 'Feshbach resonances in the presence of inelastic scattering: threshold behavior and suppression of poles in scattering lengths.' New J. Phys. **9**, 152 (2007).

[15] B. R. Johnson. 'The renormalized Numerov method applied to calculating bound states of the coupled-channel Schrödinger equation.' J. Chem. Phys. **69**, 4678 (1978).

[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran*, (Cambridge University Press, 1992), chap. 9, pp. 352–355.

[17] A. E. Thornley and J. M. Hutson. 'Bound-state wavefunctions from coupled-channel calculations using log-derivative propagators — application to spectroscopic intensities in Ar-HF.' J. Chem. Phys. **101**, 5578 (1994).

[18] A. M. Arthurs and A. Dalgarno. 'The theory of scattering by a rigid rotator.' Proc. Roy. Soc., Ser. A **256**, 540 (1960).

[19] S. Green. 'Vibrational dependence of pressure induced spectral linewidths and line shifts: Application of the infinite order sudden scattering approximation.' J. Chem. Phys. **70**, 4686 (1979).

[20] S. Green. 'Rotational excitation in H_2–H_2 collisions — close-coupling calculations.' J. Chem. Phys. **62**, 2271 (1975).

[21] S. Green. 'Comment of fitting *ab initio* intermolecular potentials for scattering calculations.' J. Chem. Phys. **67**, 715 (1977).

[22] T. G. Heil, S. Green, and D. J. Kouri. 'The coupled states approximation for scattering of two diatoms.' J. Chem. Phys. **68**, 2562 (1978).

[23] T. R. Phillips, S. Maluendes, and S. Green. 'Collision dynamics for an asymmetric top rotor and a linear rotor: Coupled channel formalism and application to H_2O–H_2.' J. Chem. Phys. **102**, 6024 (1995).

[24] S. Green. 'Rotational excitation of symmetric top molecules by collisions with atoms: Close coupling, coupled states, and effective potential calculations for NH_3–He.' J. Chem. Phys. **64**, 3463 (1976).

[25] S. Green. 'Rotational excitation of symmetric top molecules by collisions with atoms. 2. Infinite-order sudden approximation.' J. Chem. Phys. **70**, 816 (1979).

[26] J. M. Hutson and A. E. Thornley. 'Atom-spherical top van der Waals complexes: A theoretical study.' J. Chem. Phys. **100**, 2505 (1994).

[27] J. M. Hutson and F. R. McCourt. 'Close-coupling calculations of transport and relaxation cross sections for H_2 in Ar.' J. Chem. Phys. **80**, 1135 (1984).

[28] G. Wolken Jr. 'Theoretical studies of atom-solid elastic scattering: He + LiF.' J. Chem. Phys. **58**, 3047 (1973).

[29] J. M. Hutson and C. Schwartz. 'Selective adsorption resonances in the scattering of helium atoms from xenon coated graphite — close-coupling calculations and potential dependence.' J. Chem. Phys. **79**, 5179 (1983).

[30] H. Rabitz. 'Effective potentials in molecular collisions.' J. Chem. Phys. **57**, 1718 (1972).

[31] P. McGuire and D. J. Kouri. 'Quantum mechanical close coupling approach to molecular collisions. j_z-conserving coupled states approximation.' J. Chem. Phys. **60**, 2488 (1974).

[32] S. Green. 'Accuracy of decoupled–L-dominant approximation for atom-molecule scattering.' J. Chem. Phys. **65**, 68 (1976).

[33] A. E. DePristo and M. H. Alexander. 'Decoupled L-dominant approximation for ion-molecule and atom-molecule collisions.' J. Chem. Phys. **64**, 3009 (1976).

[34] R. Goldflam, D. J. Kouri, and S. Green. 'On the factorization and fitting of molecular scattering information.' J. Chem. Phys. **67**, 5661 (1977).

[35] G. A. Parker and R. T Pack. 'Rotationally and vibrationally inelastic scattering in the rotational IOS approximation. ultrasimple calculation of total (differential, integral, and transport) cross sections for nonspherical molecules.' J. Chem. Phys. **68**, 1585 (1978).

[36] R. de Vogelaere. 'A method for the numerical integration of differential equations of 2nd order without explicit 1st derivatives.' J. Res. Natl. Bur. Stand. **54**, 119 (1955).

[37] E. B. Stechel, R. B. Walker, and J. C. Light. 'R-matrix solution of coupled equations for inelastic scattering.' J. Chem. Phys. **69**, 3518 (1978).

[38] B. R. Johnson. 'Multichannel log-derivative method for scattering calculations.' J. Comput. Phys. **13**, 445 (1973).

[39] D. E. Manolopoulos, M. J. Jamieson, and A. D. Pradhan. 'Johnson's log derivative algorithm rederived.' J. Comput. Phys. **105**, 169 (1993).

[40] D. E. Manolopoulos. 'An improved log-derivative method for inelastic scattering.' J. Chem. Phys. **85**, 6425 (1986).

[41] D. E. Manolopoulos. *Close-coupled equations: the log-derivative approach to inelastic scattering, bound-state and photofragmentation problems*. Ph.D. thesis, Cambridge University, Cambridge (1988).

[42] M. H. Alexander. 'Hybrid quantum scattering algorithms for long-range potentials.' J. Chem. Phys. **81**, 4510 (1984).

[43] M. H. Alexander and D. E. Manolopoulos. 'A stable linear reference potential algorithm for solution of the quantum close-coupled equations in molecular scattering theory.' J. Chem. Phys. **86**, 2044 (1987).

[44] G. A. Parker, T. G. Schmalz, and J. C. Light. 'A variable interval variable step method for the solution of linear 2nd order coupled differential-equations.' J. Chem. Phys. **73**, 1757 (1980).

[45] R. T Pack. 'Space-fixed vs body-fixed axes in atom-diatomic molecule scattering. Sudden approximations.' J. Chem. Phys. **60**, 633 (1974).

[46] W. M. Huo and S. Green. 'Quantum calculations for rotational energy transfer in nitrogen molecule collisions.' J. Chem. Phys. **104**, 7572 (1996).

[47] R. Goldflam, S. Green, D. J. Kouri, and L. Monchick. 'Effect of molecular anisotropy on beam scattering measurements.' J. Chem. Phys. **69**, 598 (1978).

[48] J. L. Bohn, M. Cavagnero, and C. Ticknor. 'Quasi-universal dipolar scattering in cold and ultracold gases.' New J. Phys. **11**, 055039 (2009).

[49] T. Karman, M. D. Frye, J. D. Reddel, and J. M. Hutson. 'Near-threshold bound states of the dipole-dipole interaction.' Phys. Rev. A **98**, 062502 (2018).

[50] M. L. González-Martínez and J. M. Hutson. 'Ultracold atom-molecule collisions and bound states in magnetic fields: zero-energy Feshbach resonances in He-NH ($^3\Sigma^-$).' Phys. Rev. A **75**, 022702 (2007).

[51] G. A. Parker, J. C. Light, and B. R. Johnson. 'The logarithmic derivative - variable interval variable step hybrid method for the solution of coupled linear 2nd-order differential-equations.' Chem. Phys. Lett. **73**, 572 (1980).

[52] T. Karman, L. M. C. Janssen, R. Sprenkels, and G. C. Groenenboom. 'A renormalized potential-following propagation algorithm for solving the coupled-channels equations.' J. Chem. Phys. **141**, 064102 (2014).

[53] J. M. Hutson and B. J. Howard. 'Spectroscopic properties and potential surfaces for atom-diatom van der Waals molecules.' Mol. Phys. **41**, 1123 (1980).

[54] R. Shafer and R. G. Gordon. 'Quantum scattering theory of rotational relaxation and spectral line shapes in H_2-He gas mixtures.' J. Chem. Phys. **58**, 5422 (1973).

[55] G. Fisanick-Englot and H. Rabitz. 'Studies of inelastic molecular collisions using impact parameter methods. III. Line shape functions.' J. Chem. Phys. **63**, 1547 (1975).

[56] G. Goldflam and D. J. Kouri. 'On accurate quantum mechanical approximations for molecular relaxation phenomena. averaged j_z-conserving coupled states approximation.' J. Chem. Phys. **66**, 542 (1977).

[57] S. Green, L. Monchick, G. Goldflam, and D. J. Kouri. 'Computational tests of angular momentum decoupling approximations for pressure broadening cross sections.' J. Chem. Phys. **66**, 1409 (1977).

[58] R. Goldflam, S. Green, and D. J. Kouri. 'Infinite order sudden approximation for rotational energy transfer in gaseous mixtures.' J. Chem. Phys. **67**, 4149 (1977).

[59] R. Blackmore, S. Green, and L. Monchick. 'Polarized D_2 Stokes-Raman Q-branch broadened by He: A numerical calculation.' J. Chem. Phys. **88**, 4113 (1988).

[60] S. Green. 'Rotational excitation in collisions between two rigid rotors: Alternate angular momentum coupling and pressure broadening of HCl by H_2.' Chem. Phys. Lett. **47**, 119 (1977).

[61] S. Green. 'Rotational excitation of symmetric top molecules by collisions with atoms. ii. infinite order sudden approximation.' J. Chem. Phys. **70**, 816 (1979).

[62] C. J. Ashton, M. S. Child, and J. M. Hutson. 'Rotational predissociation of the Ar-HCl van der Waals complex - close-coupled scattering calculations.' J. Chem. Phys. **78**, 4025 (1983).

[63] W. Brenig and R. Haag. 'Allgemeine Quantentheorie der Stoßprozesse.' Fortschr. Phys. **7**, 183 (1959).

[64] J. R. Taylor. *Scattering Theory: The Quantum Theory of Nonrelativistic Collisions*. (Wiley, New York, 1972).

[65] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran*, (Cambridge University Press, 1992), chap. 4, p. 128.

[66] J. M. Hutson. 'Coupled channel bound state calculations: Calculating expectation values without wavefunctions.' Chem. Phys. Lett. **151**, 565 (1988).

[67] J. F. E. Croft, A. O. G. Wallis, J. M. Hutson, and P. S. Julienne. 'Multichannel quantum defect theory for cold molecular collisions.' Phys. Rev. A **84**, 042703 (2011).

[68] J. F. E. Croft, J. M. Hutson, and P. S. Julienne. 'Optimized multichannel quantum defect theory for cold molecular collisions.' Phys. Rev. A **86**, 022711 (2012).

[69] J. M. Hutson. 'Vibrational dependence of the anisotropic intermolecular potential of Ar-HF.' J. Chem. Phys. **96**, 6752 (1992).

[70] J. M. Hutson, A. Ernesti, M. M. Law, C. F. Roche, and R. J. Wheatley. 'The intermolecular potential energy surface for CO_2-Ar: Fitting to high-resolution spectroscopy of Van der Waals complexes and second virial coefficients.' J. Chem. Phys. **105**, 9130 (1996).

[71] C. F. Roche, A. S. Dickinson, A. Ernesti, and J. M. Hutson. 'Line shape, transport and relaxation properties from intermolecular potential energy surfaces: The test case of CO_2-Ar.' J. Chem. Phys. **107**, 1824 (1997).

[72] R. J. Le Roy and J. S. Carley. 'Spectroscopy and potential energy surfaces of van der Waals molecules.' Adv. Chem. Phys. **42**, 353 (1980).

[73] G. Zarur and H. Rabitz. 'Effective potential formulation of molecule-molecule collisions with application to H_2–H_2.' J. Chem. Phys. **60**, 2057 (1974).

[74] U. Buck, A. Kohlhase, T. Phillips, and D. Secrest. 'Differential energy-loss spectra for CH_4 + Ar collisions.' Chem. Phys. Lett. **98**, 199 (1983).

[75] W. B. Chapman, A. Schiffman, J. M. Hutson, and D. J. Nesbitt. 'Rotationally inelastic scattering in CH_4 + He, Ne, and Ar: State-to-state cross sections via direct infrared laser absorption in crossed supersonic jets.' J. Chem. Phys. **105**, 3497 (1996).

[76] P. Soldán, P. S. Żuchowski, and J. M. Hutson. 'Prospects for sympathetic cooling of polar molecules: NH with alkali-metal and alkaline-earth atoms — a new hope.' Faraday Discuss. **142**, 191 (2009).

[77] A. O. G. Wallis and J. M. Hutson. 'Production of ultracold NH molecules by sympathetic cooling with Mg.' Phys. Rev. Lett. **103**, 183201 (2009).

[78] C. Strauss, T. Takekoshi, F. Lang, K. Winkler, R. Grimm, J. Hecker Denschlag, and E. Tiemann. 'Hyperfine, rotational, and vibrational structure of the $a^3\Sigma_u^+$ state of $^{87}Rb_2$.' Phys. Rev. A **82**, 052514 (2010).

[79] J. M. Hutson. 'Vibrational dependence of the anisotropic intermolecular potential of Ar-HCl.' J. Phys. Chem. **96**, 4237 (1992).

[80] J. M. Hutson, C. J. Ashton, and R. J. Le Roy. 'Vibrational predissociation of H_2-Ar, D_2-Ar, and HD-Ar van der Waals molecules.' J. Phys. Chem. **87**, 2713 (1983).

[81] R. J. Le Roy and J. M. Hutson. 'Potential energy surfaces for H_2 with Ar, Kr and Xe.' J. Chem. Phys. **86**, 837 (1987).

[82] E. Arimondo, M. Inguscio, and P. Violino. 'Experimental determinations of the hyperfine structure in the alkali atoms.' Rev. Mod. Phys. **49**, 31 (1977).

[83] J. M. Hutson, E. Tiesinga, and P. S. Julienne. 'Avoided crossings between bound states of ultracold cesium dimers.' Phys. Rev. A **78**, 052703 (2008). Note that the matrix element given in Eq. A2 of this paper omits a factor of $-\sqrt{30}$ and that for Eq. A5 omits a factor of $\frac{1}{2}$.

[84] J. M. Hutson and B. J. Howard. 'Anisotropic intermolecular forces. I. Rare gas-hydrogen chloride systems.' Mol. Phys. **45**, 769 (1982).

[85] J. M. Hutson. 'The intermolecular potential of Ar–HCl: Determination from high-resolution spectroscopy.' J. Chem. Phys. **89**, 4550 (1988).

[86] J. M. Hutson. 'The intermolecular potential of Ne–HCl: determination from high-resolution spectroscopy.' J. Chem. Phys. **91**, 4448 (1989).

[87] J. M. Hutson. 'Intermolecular potential energy surfaces for Kr–HCl and Ar–HBr.' J. Chem. Phys. **91**, 4455 (1989).

[88] T. S. Ho and H. Rabitz. 'A general method for constructing multidimensional molecular potential energy surfaces from *ab initio* calculations.' J. Chem. Phys. **104**, 2584 (FEB 1996).

[89] P. Soldán and J. M. Hutson. 'On the long-range and short-range behavior of potentials from reproducing kernel hilbert space interpolation.' J. Chem. Phys. **112**, 4415 (MAR 2000).

[90] B. M. Smirnov and M. I. Chibisov. 'Electron exchange and changes in hyperfine state of colliding alkaline metal atoms.' Sov. Phys. JETP **21**, 624 (1965).

Index of key variables and arrays