

MOLSCAT, BOUND and FIELD

Version 2019.0 (beta)

User Manual

Jeremy M. Hutson and C. Ruth Le Sueur

November 12, 2018

Contents

Contents	1
1 Preamble	1
1.1 Citing the programs	1
1.2 Licensing	1
1.3 Scope of the programs	1
1.4 Summary of program capabilities	2
1.5 Structure of documentation	4
1.6 Program history	5
1.7 Principal changes in version 2019.0 (beta)	6
2 Theory	8
2.1 Convention for quantum numbers	9
2.2 Matrix of the interaction potential	10
2.3 Matrices of the internal and centrifugal Hamiltonians	10
2.4 Scattering calculations	11
2.4.1 Low-energy collision properties	12
2.4.2 Infinite-order sudden approximation	13
2.5 Bound-state calculations	14
3 Using the programs: a basic guide	16
3.1 Prerequisites	16
3.2 Input data format	17
3.3 Interaction types	18
3.4 Interaction potential	19
3.5 Propagators	19
3.6 Overview of input data file	20
3.6.1 Scattering and bound-state calculations	20
3.6.2 Scattering calculations	21
3.6.3 Bound-state calculations	21
3.7 Examples for MOLSCAT	22
3.7.1 Interpretation of the complete output for a model system	22
3.7.2 Higher print level	26
3.8 Example for BOUND	31
3.9 Calculations in external fields	33
3.9.1 Using MOLSCAT to calculate the field-dependent scattering length . . .	33

3.9.2	Using FIELD to identify threshold crossings	34
3.9.3	Using BOUND and FIELD to build the bound-state picture	35
4	Interaction types and basis sets	36
4.1	Interaction types	36
4.2	Pair levels, pair states and pair basis functions	37
4.2.1	Basis sets diagonal in H_{intl} and \hat{L}^2	37
4.2.2	Basis sets non-diagonal in H_{intl} or \hat{L}^2	38
4.3	Units of energy	39
4.4	Specification of pair level quantum numbers and energies	39
4.4.1	Linear rigid rotor + atom (ITYP = 1)	40
4.4.2	Diatomic vibrotor + atom (ITYP = 2 and 7)	40
4.4.3	Linear rigid rotor + linear rigid rotor (ITYP = 3)	41
4.4.4	Rigid symmetric (or near-symmetric) top + atom (ITYP = 5)	42
4.4.5	Asymmetric (or spherical) top + atom (ITYP = 6)	43
4.4.6	Asymmetric rigid rotor + linear rigid rotor (ITYP = 4)	46
4.4.7	Atom + rigid corrugated surface (ITYP = 8)	46
4.5	Additional quantum numbers in JSTATE but not JLEVEL	48
4.6	Close-coupling calculations	48
4.7	Decoupling approximations	49
4.7.1	Effective potential	49
4.7.2	CS (Coupled-states / centrifugal sudden) and helicity decoupling	49
4.7.3	Decoupled L -Dominant	50
4.8	Loops over JTOT and IBLOCK	50
4.9	Contracting the basis set (BOUND only)	51
4.10	IOS calculations (MOLSCAT only, IADD = 100)	51
4.11	The BCT Hamiltonian	51
4.12	Plug-in basis-set suites (ITYPE = 9)	52
5	Constructing the interaction potential	53
5.1	The potential expansion	53
5.1.1	Rigid rotor + atom (ITYP = 1)	54
5.1.2	Diatomic vibrotor + atom (ITYP = 2 or 7)	55
5.1.3	Rigid rotor + rigid rotor (ITYP = 3)	55
5.1.4	Non-linear molecule + atom (ITYP = 5 or 6)	56
5.1.5	Asymmetric rotor + diatom (ITYP = 4)	56
5.1.6	Atom + corrugated solid surface (ITYP = 8)	57
5.2	Units of length and energy for the interaction potential	57
5.3	Evaluating the radial potential coefficients	58
5.3.1	Potential supplied by VINIT/VSTAR pre-expanded in internal coordinates	58
5.3.2	Potential supplied by VRTP as function of internal coordinates	59
5.4	Module angles	61
5.5	IOS calculations (MOLSCAT only, IADD = 100)	62
6	Specifying input energies	63

6.1	Energy units	63
6.2	Specifying energies for MOLSCAT and FIELD calculations	63
6.3	Internally generated energies in MOLSCAT	64
6.4	Temperatures for thermal averaging (MOLSCAT only)	64
6.5	Specifying energies for BOUND calculations	64
6.6	The reference energy	65
6.6.1	Specifying the index of the reference threshold	65
7	External fields and scaling the interaction potential	66
7.1	Using external fields	67
7.2	Potential scaling	67
8	Controlling the propagators	69
8.1	Propagator choice	69
8.1.1	Units of length	70
8.1.2	Ranges of integration	70
8.1.3	Step size	71
8.1.4	3-segment propagation in BOUND and FIELD	72
8.2	Convergence of propagations	73
8.3	Propagators in more detail	73
8.3.1	de Vogelaere propagator (propagator 02) (MOLSCAT only)	73
8.3.2	R-matrix propagator (propagator 03) (MOLSCAT only)	73
8.3.3	VIVS propagator (propagator 14) (MOLSCAT only)	74
8.3.4	LDJ, LDMD and LDMA propagators (propagators 05, 06 and 07)	75
8.3.5	LDMG propagator (propagator 9)	76
8.3.6	AIRY propagator (propagator 01)	76
8.3.7	WKB integration (propagator -1) (MOLSCAT only)	77
8.4	Boundary conditions for log-derivative propagators	77
8.5	Internal units of length and energy	79
8.6	Propagator scratch file	79
9	Controlling scattering calculations (MOLSCAT only)	80
9.1	Asymptotic basis sets	80
9.2	Resolving degeneracies with extra operators	80
9.3	Channel indices for open channels	81
9.4	S matrix	81
9.5	Automated convergence testing	81
9.6	Elastic and inelastic cross sections	82
9.6.1	Partial cross sections	83
9.6.2	Restarting cross section calculations	83
9.6.3	Integral cross sections	83
9.7	Line-shape cross sections	84
9.8	Calculating line-shape cross sections within the IOS approximation	85
9.9	Locating scattering resonances as a function of energy	85
9.10	Calculating low-energy collision properties	87
9.10.1	Scattering lengths/volumes	87

9.10.2	Scanning the scattering length/volume over external field	87
9.10.3	Characterising a resonance in the scattering length as a function of an external field	87
9.10.4	Converging on a specific value of the scattering length/volume	88
9.10.5	Effective range	88
9.11	Calculating scattering wavefunction	89
10	Controlling bound-state calculations (BOUND and FIELD only)	90
10.1	The node count	90
10.2	Locating bound states with BOUND	90
10.3	Locating bound states with FIELD	91
10.4	Bound-state wavefunctions	91
10.5	Calculating expectation values (BOUND only)	91
10.6	Automatic convergence testing (BOUND only)	92
10.7	Richardson extrapolation to zero step size (BOUND only)	93
11	Complete list of input parameters	94
11.1	Items in namelist &INPUT	94
11.2	Items in namelist &BASIS	101
11.3	Items in namelist &POTL	104
12	Controlling the print level	107
12.1	Levels for IPRINT	107
12.2	Basis sets and quantum numbers	108
12.3	Coupling matrices	108
12.4	Energies and channel indices	108
12.4.1	Calculation of threshold energies for non-diagonal basis sets	109
12.4.2	Resolving degeneracies amongst threshold energies	109
12.5	Propagation(s)	109
12.6	Scattering calculations (MOLSCAT only)	110
12.6.1	S-matrix elements	110
12.6.2	Convergence of S-matrix elements	111
12.6.3	Cross sections (non- IOS calculations)	111
12.6.4	Cross sections (IOS only)	111
12.6.5	Line-shape cross sections	111
12.6.6	Scattering lengths/volumes	112
12.6.7	Location of energy-dependent resonances in the eigenphase sum	112
12.6.8	Characterisation of field-dependent resonances	112
12.6.9	Location of field at which the scattering length/volume has specific value	112
12.6.10	Effective range	112
12.6.11	State-to-state cross sections in main output file	112
12.7	Bound-state calculations (BOUND and FIELD only)	113
12.7.1	Location of bound states	113
12.7.2	Expectation values (BOUND only)	113
12.7.3	Convergence testing (MOLSCAT and BOUND only)	113

12.7.4	Calculation of wavefunction	113
13	Auxiliary output and scratch files	114
13.1	Summary of bound states (BOUND and FIELD only)	115
13.2	Summary of eigenphase sums and low-energy scattering lengths (MOLSCAT only)	115
13.3	State-to-state integral cross sections (MOLSCAT only)	115
13.4	Partial cross sections (MOLSCAT only)	115
13.5	S and K matrices (MOLSCAT only)	115
13.5.1	S matrices (MOLSCAT only)	116
13.5.2	K matrices (MOLSCAT only)	118
13.6	Wavefunctions (LDMD propagator only)	119
13.7	Log-derivative matrices (MOLSCAT only)	119
13.8	Coupling matrices	120
14	Example input and output files	121
14.1	Examples for MOLSCAT	121
14.1.1	All available propagators for MOLSCAT	121
14.1.2	All available coupling approximations (using ITYP = 2)	122
14.1.3	Location of quasibound state (Feshbach resonance) for Ar-HF	122
14.1.4	Line-shape cross sections for Ar + CO ₂	122
14.1.5	Line-shape cross sections for Ar + H ₂	123
14.1.6	ITYP = 3: Cross sections for rigid rotor + rigid rotor collisions	123
14.1.7	ITYP = 5: cross sections for atom + symmetric top collisions, illustrating automated convergence testing	123
14.1.8	ITYP = 6: Cross sections for atom + spherical top collisions	124
14.1.9	ITYP = 8: Atom-surface scattering	124
14.1.10	ITYP = 9: Cross sections for Mg+NH in a magnetic field	124
14.1.11	ITYP = 9: Characterisation of magnetically tunable Feshbach resonances and calculation of effective range for ⁸⁵ Rb+ ⁸⁵ Rb	125
14.2	Examples for BOUND	126
14.2.1	All available propagators for bound-state calculations	126
14.2.2	Bound states of Ar-HCl with expectation values	126
14.2.3	Bound states of Ar-CO ₂ with Richardson extrapolation	126
14.2.4	Bound states of Ar-H ₂	127
14.2.5	Bound states of H ₂ -H ₂ (ortho-para)	127
14.2.6	Bound states of He-NH ₃	127
14.2.7	Bound states of Ar-CH ₄	127
14.2.8	Bound-state energies of Mg + NH at specified magnetic fields	128
14.3	Examples for FIELD	128
14.3.1	Bound states of Mg + NH as a function of magnetic field	128
14.3.2	Locating threshold crossings for ⁸⁵ Rb ₂	128
14.3.3	Bound states of ⁸⁵ Rb ₂ as a function of magnetic field	129
15	Installing and testing the programs	130
15.1	Supplied files	130

15.2	Program language	131
15.3	Main program	131
15.4	Integer length	131
15.5	Date, time and CPU time routines	131
15.6	Linear algebra	131
15.7	Matrix multiplication	132
15.8	Symmetric matrix inversion	132
15.9	Linear equation solver	132
15.10	Eigenvalues and eigenvectors of symmetric matrices	133
15.11	File handling	133
15.12	Values of fundamental constants	133
15.13	Array dimensioning	133
15.14	Dimensions of fixed-size arrays	134
15.15	COMMON blocks	135
15.16	Testing the installation	136
16	Plug-in potential routine (POTENL)	137
16.1	Specification of POTENL subroutine	137
16.1.1	Initialisation	138
16.1.2	Evaluation (IC=0, 1 or 2)	139
17	Plug-in basis-set suites	140
17.1	Components of a basis-set suite	140
17.2	Diagonal or non-diagonal Hamiltonian	141
17.3	Interpretation of external loop variables	142
17.4	Calculating the interaction matrix	143
17.5	Routine BAS9IN	143
17.6	Routine SET9	145
17.7	Routine BASE9	147
17.8	Routine POTIN9	147
17.9	Routine CPL9	149
17.10	Additional subroutines required in some cases	150
17.10.1	DEGEN9 : denominators for degeneracy-averaged cross sections	150
17.10.2	THRSH9 : threshold energies from monomer quantum numbers	151
17.10.3	EFV9 : Converting EFVs to values in the VCONST array	151
17.11	Resolving threshold degeneracies with extra operators	152
17.12	Modules available for use in plug-in basis-set suites	152
17.12.1	Module basis_data	152
17.12.2	Module potential	153
17.12.3	Module efvs	153
18	Supplied plug-in basis-set suites	155
18.1	1S atom + $^3\Sigma$ diatom	155
18.1.1	Additional information for programmers	156
18.2	Alkali-metal atom + alkali-metal atom	157
18.2.1	Additional information for programmers	159

18.2.2	Extra operator functionality	160
19	Supplied potential routines	161
19.1	Potential energy surfaces for Ar-HF and Ar-HCl	161
19.2	Potential curves in the form of Tiemann and coworkers	161
20	Bibliography	164

Chapter 1

Preamble

1.1 Citing the programs

Any publication that uses MOLSCAT, BOUND or FIELD should cite them as:

Jeremy M. Hutson and C. Ruth Le Sueur, MOLSCAT: a program for non-reactive quantum scattering calculation on atomic and molecular collisions, Version 2019.0 (beta)

Jeremy M. Hutson and C. Ruth Le Sueur, BOUND: a program for bound states of interacting pairs of atoms and molecules, Version 2019.0 (beta)

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 2019.0 (beta)

or refer to the published paper in Computer Physics Communications when available.

1.2 Licensing

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. 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 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 fields, 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.

Previous versions of these programs have been 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.

1.4 Summary of 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.

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. Two such suites are included in this distribution, for

- ^1S 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 fields, the reference energy may be 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 towards an energy where there is a scattering resonance (predissociating / quasibound state of the collision complex)
- Output K matrices to files for subsequent processing to characterize scattering resonances;
- Calculate low-energy scattering properties such as scattering lengths, scattering volumes and effective ranges;
- Converge on and characterize low-energy Feshbach resonances in the scattering length as a function of external field.

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 fields, 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 bound states that exist in a specified range of external fields at a specified list or grid of energies.

1.5 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.

1.6 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 Web site. 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 has been no fully documented publication of the programs since MOLSCAT version 14 and BOUND version 5, and FIELD has never been published. However, there have been many enhancements to the capabilities in the intervening years, and updates have been provided privately to selected users. To place the enhancements in a chronological sequence, we refer to these unpublished versions in some places as “version 2007.1”, etc. Nevertheless, we treat all changes since MOLSCAT version 14 and BOUND version 5 as new in the present version, since this is the first time they have been collected together and documented.

1.7 Principal changes in version 2019.0 (beta)

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

This functionality has been 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 transforms the propagated wavefunction or log-derivative matrix to a basis set that is diagonal in H_{intl} before matching to long-range functions to extract the S matrix.

MOLSCAT now processes the S matrix to calculate scattering lengths (or volumes or hypervol-

umes) a_L [actually k -dependent complex scattering lengths/volumes $a_L(k)$] for any low-energy scattering channels. It can also extrapolate the real part of $a_0(k)$ to $k = 0$ and calculate the effective range r_{eff} .

MOLSCAT can 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 can calculate a multichannel scattering wavefunction for flux incoming in a single channel.

BOUND and FIELD now separate R_{mid} , the distance where the calculation switches between short-range and long-range propagators, and R_{match} , the distance where the incoming and outgoing wavefunctions are matched.

BOUND and FIELD now do an outwards propagation from R_{min} to R_{match} and an inwards propagation from R_{max} to R_{match} . The node count is calculated without needing a third propagation from R_{match} to R_{min} or R_{max} .

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

All the programs now allow more general choice of log-derivative boundary conditions at the starting points for propagation.

A new propagation approach [9] has been 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 channel equations. Coefficients for two SIs are 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 has been coded so that other SIs could easily be implemented if desired.

Chapter 2

Theory

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

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

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,

$$H_{\text{intl}}(\xi_{\text{intl}}) = H_{\text{intl}}^{(1)}(\xi_{\text{intl}}^{(1)}) + H_{\text{intl}}^{(2)}(\xi_{\text{intl}}^{(2)}), \quad (2.2)$$

with eigenvalues $E_{\text{intl},i} = E_{\text{intl},i}^{(1)} + E_{\text{intl},i}^{(2)}$, where $E_{\text{intl},i}^{(1)}$ and $E_{\text{intl},i}^{(2)}$ 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.

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

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

where the functions $\Phi_j(\xi_{\text{intl}})$ form a complete orthonormal basis set for motion in the coordinates ξ_{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 $\Phi_i(\xi_{\text{intl}})$. The resulting coupled differential equations for the channel functions $\psi_i(R)$ are

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

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

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

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,

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

If there are N basis functions included in the expansion (2.3), $\psi(R)$ 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 $\psi(R)$ that satisfy the Schrödinger equation subject to the boundary condition that $\psi(R) \rightarrow 0$ in the classically forbidden region at short range. These N column vectors form a wavefunction matrix $\Psi(R)$. The various propagators in MOLSCAT, BOUND and FIELD work either by propagating $\Psi(R)$ and its radial derivative $\Psi'(R)$ or by propagating the log-derivative matrix $\mathbf{Y}(R) = \Psi'(R)[\Psi(R)]^{-1}$.

The particular choice of the basis functions $\Phi_j(\xi_{\text{intl}})$ 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_{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.

2.1 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 .

2.2 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,

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

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 \mathcal{V}^{Λ} with elements $\mathcal{V}_{ij}^{\Lambda} = \langle \Phi_i | \mathcal{V}^{\Lambda} | \Phi_j \rangle_{\text{intl}}$ are calculated once and stored for use in evaluating $W_{ij}(R)$ throughout the course of a propagation.

2.3 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 $\langle \Phi_i | H_{\text{intl}} | \Phi_j \rangle_{\text{intl}} = E_{\text{intl},i} \delta_{ij}$. 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

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

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

$$\hat{L}^2 = \sum_{\Upsilon} \mathcal{L}^{\Upsilon}. \quad (2.9)$$

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 .

2.4 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 $R \rightarrow \infty$ ($E_{\text{intl},i} \leq E$) 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 $\Psi(R)$ 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,

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

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. 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,

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

where \mathbf{K}_{oo} is the open-open portion of the K matrix. \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 $\Psi(R)$ or its log-derivative $\mathbf{Y}(R)$.

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

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

Finally MOLSCAT uses the transformed $\Psi(R_{\text{max}})$ or $\mathbf{Y}(R_{\text{max}})$, together with the eigenvalues $E_{\text{intl},i}$ 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*,¹ with contributions from many values of J_{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

$$\sigma_{n_i \rightarrow n_f} = \frac{\pi}{g_{n_i} k_{n_i}^2} \sum_{\substack{J_{\text{tot}} \\ M}} (2J_{\text{tot}} + 1) \sum_{\substack{i \in n_i \\ f \in n_f}} \left| \delta_{if} - S_{if}^{J_{\text{tot}}, M} \right|^2. \quad (2.12)$$

Here n_i and n_f label initial and final levels (not states) of the colliding pair,² while i and f indicate the open channels arising from those levels for total angular momentum J_{tot} and symmetry block M . g_{n_i} is the degeneracy of level n_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, and use it to converge towards a far-off scattering resonance as a function of energy.

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

- 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.

2.4.1 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

¹Note 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.

²For some interaction types, n_i and n_f each represent several quantum numbers, not just one.

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,

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

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 [12]

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

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 $k \rightarrow 0$. 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 [12].

2.4.2 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 dynamical factors Q multiplied by 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.

2.5 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, $E < E_{\text{intl},i}$ for all i . Under these circumstances the bound-state wavefunction $\psi(R)$ is a column vector of order N that must approach zero in the classically forbidden regions at both short range, $R \rightarrow 0$, and long range, $R \rightarrow \infty$.

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 $\Psi(R)$ that satisfy the boundary conditions in *one* of the classically forbidden regions exist at any energy. We designate these $\Psi^+(R)$ for the solution propagated outwards from short range and $\Psi^-(R)$ 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

$$\psi(R_{\text{match}}) = \psi^+(R_{\text{match}}) = \psi^-(R_{\text{match}}). \quad (2.15)$$

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

$$\frac{d}{dR}\psi^+(R_{\text{match}}) = \frac{d}{dR}\psi^-(R_{\text{match}}) \quad (2.16)$$

so that

$$\mathbf{Y}^+(R_{\text{match}})\psi(R_{\text{match}}) = \mathbf{Y}^-(R_{\text{match}})\psi(R_{\text{match}}). \quad (2.17)$$

Equivalently,

$$[\mathbf{Y}^+(R_{\text{match}}) - \mathbf{Y}^-(R_{\text{match}})]\psi(R_{\text{match}}) = 0, \quad (2.18)$$

so that the wavefunction vector $\psi(R_{\text{match}})$ is an eigenvector of the log-derivative matching matrix, $\Delta\mathbf{Y} = [\mathbf{Y}^+(R_{\text{match}}) - \mathbf{Y}^-(R_{\text{match}})]$, 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 [14], which is defined as the number of eigenvalues of the coupled equations that lie below E . It uses the node count to determine the number of eigenvalues of the coupled equations in the specified range, and then uses bisection to identify energy windows that contain exactly one eigenvalue. In each such window, it uses a combination of bisection and the Van Wijngaarden-Dekker-Brent algorithm [15] to converge on

the energy where an eigenvalue of the log-derivative matching matrix $\Delta \mathbf{Y}$ is zero. This energy is an eigenvalue of the coupled equations. The program extracts the local wavefunction vector $\psi(R_{\text{match}})$, and optionally calculates the complete bound-state wavefunction $\psi(R)$ using the method of Thornley and Hutson [16].

FIELD operates in a very similar manner to locate eigenvalues of the coupled equations 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.

The choice of R_{match} is significant. It does not affect the energies or fields at which the matching condition (2.18) is satisfied, but it does affect the value of the matching determinant at other energies or fields and hence the rate of convergence on eigenvalues. In particular, it is usually inappropriate to place R_{match} far into a classically forbidden region.

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 $\{\Phi_j(\xi_{\text{intl}})\}$ to be used for the internal coordinates ξ_{intl} (everything except the interparticle separation R).
- the interaction potential $V(R, \xi_{\text{intl}})$;
- 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 data 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 data 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, integration 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 [17];
2. Atom + vibrating diatom (rotationally and/or vibrationally inelastic) with interaction potentials independent of diatom rotational state [18];
3. Linear rigid rotor + linear rigid rotor [19, 20, 21];
4. Asymmetric top + linear molecule [22]
5. Atom + symmetric top (also handles near-symmetric tops and linear molecules with vibrational angular momentum) [23, 24];
6. Atom + asymmetric top [23] (also handles spherical tops [25]);
7. Atom + vibrating diatom (rotationally and/or vibrationally inelastic) with interaction potentials dependent on diatom rotational state [26];
8. Atom + rigid corrugated surface: diffractive (elastic) scattering [27, 28]. 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 $^{85}\text{Rb}_2$.

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 [29] (seldom used nowadays);
- Coupled-states (centrifugal sudden) approximation [30];
- Decoupled L -dominant approximation [31, 32] (seldom used nowadays);
- Infinite-order sudden approximation (MOLSCAT only) [33, 34, 24].

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 the 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:

AIRY propagator: This is the AIRY log-derivative propagator of Alexander [35] as reformulated by Alexander and Manolopoulos [36]. It uses a locally adiabatic 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.

de Vogelaere propagator (DV) [37]: This is reliable and accurate, 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) [38]: This is a stable method that works in a quasiadiabatic basis. It is simple to use, but has relatively poor step-size convergence properties, so that it is not generally suitable for obtaining very high accuracy. It has largely been superseded by the log-derivative propagators.

Log-derivative propagator of Johnson (LDJ) [39]: This is a very stable and reliable propagator, which is particularly efficient on vector processors. It has now largely been superseded by the LDMD propagator, but can be useful in occasional cases where that propagator has trouble with node counts.

Diabatic log-derivative method of Manolopoulos (LDMD) [40, 41]: 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) [40]: This propagator implements the imbedding method of Manolopoulos in a quasiadiabatic basis. It offers better accuracy than the LDMD propagator for very strongly coupled problems, but is relatively expensive (even at subsequent energies). 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 coupling matrix at specific distances (adiabats).

Symplectic log-derivative propagators of Manolopoulos and Gray (LDMG) [9]: This code offers a choice of 4th-order or 5th-order symplectic propagators. These are 1.5–3 times more expensive per step than the LDMD and LDJ propagators, but can have smaller errors for a given step size. For smooth potential functions, they can be the most efficient choice when high precision is required. The current authors have not used them extensively, but George McBane (mcbane@gvsu.edu) is willing to provide support.

VIVS propagator [42]: This method uses the variable-interval variable-step method of Parker *et al.* and is intended for use at long range. It is commonly implemented in combination with the LDJ propagator at short range. This is sometimes the most efficient propagator if many different energies are required, but control of it is rather more complicated than for other propagators. It has largely been superseded by the AIRY propagator.

WKB semiclassical integration using Gauss-Mehler quadrature [43]: 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 potential 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 2019.0 (beta).

Recommendation:

For most practical purposes, the LDMD propagator provides the best balance between stability and efficiency at short and medium range. For problems that do not require long-range propagation, it 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 input data file

The input data 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 4.8

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 convergence testing of S-matrix elements; 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 convergence testing of bound-state energies and expectation values (BOUND only); see section 10.6
9. Specify a wavefunction calculation; see section 10.4.

3.7 Examples for MOLSCAT

3.7.1 Interpretation of the complete output for a model system

The input data 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) 2018 J. M. Hutson & C. R. Le Sueur
|
|           Version 2019.0 (beta)
|
|           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 refer to J. M. Hutson & C. R. Le Sueur, MOLSCAT computer code 2019.0 (beta) or to the published paper in Computer Physics Communications when available.

PHYSICAL CONSTANTS OBTAINED FROM NIST CODATA TABULATION OF 2014

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

PRINT LEVEL = 1 OTHER PRINT CONTROLS ISIGPR = 2

RUN LABEL = model system: ITYPE=1

REDUCED MASS FOR INTERACTION = 20.000000000 ATOMIC MASS UNITS

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.

TARGET ROTATIONAL LEVELS TAKEN FROM &BASIS (JLEVEL) INPUT.  NLEVEL =  4

ENERGY LEVELS COMPUTED FROM B(E) =   30.000000

QUANTUM NUMBERS FOR INTERACTING PAIR:
EACH PAIR STATE IS LABELLED BY  1 QUANTUM LABEL
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

      0.10000000E+01 * R ** -12
     -0.20000000E+01 * R ** -6

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

     -0.20000000E+00 * R ** -6

```

```
POTENL PROCESSING FINISHED.
```

The units used for length (“RM units”) throughout the run and for interaction energies returned from the potential subroutine may also be specified in namelist `&POTL`:

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

resulting in the following output:

```
UNITS FOR POTENTIAL ROUTINES:
POTENTIAL RETURNED IN UNITS OF EPSIL =      50.000000 CM-1
R              IN UNITS OF RM      =      3.500000 ANGSTROMS

ALL LENGTHS ARE IN UNITS OF RM =  3.500000 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 ALGORITHM FOR COUPLED EQUATIONS SPECIFIED BY IPROPS =  6

COUPLED EQUATIONS WILL BE PROPAGATED OUTWARDS IN 1 SEGMENT
+++++
SEGMENT 1 WILL BE PROPAGATED OUTWARDS

FROM RMIN CHOSEN USING IRMSET =   9  TO RMAX   =    20.00

COUPLED EQUATIONS SOLVED BY DIABATIC MODIFIED LOG DERIVATIVE METHOD OF MANOLOPOULOS

PROPAGATION STEP SIZE DETERMINED USING DR =  1.000E-03

LOG-DERIVATIVE MATRIX WILL BE INITIALISED IN THE LOCAL EIGENBASIS FOR RMIN IN THE OUTWARDS PROPAGATION PART
LOCALLY CLOSED CHANNELS WILL BE INITIALISED WITH A WKB BOUNDARY CONDITION
LOCALLY OPEN CHANNELS WILL BE INITIALISED WITH THE VALUE  1.0000000E+30
+++++
```

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

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



```

INPUT ENERGY VALUES ASSUMED TO BE IN UNITS OF CM-1 BY DEFAULT.

CONTROL DATA FOR TOTAL ENERGIES.  CALCULATIONS WILL BE PERFORMED FOR  1 VALUE
ENERGY NO.   1 =   1250.0000000    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  DIFFERENT 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 IBLOCK =  1 ENERGY( 1) = 1250.000    :  MAXIMUM DIAGONAL & OFF-DIAGONAL =  9.88E-02 &  1.16E-06
FOR JTOT =  10 IBLOCK =  2 ENERGY( 1) = 1250.000    :  MAXIMUM DIAGONAL & OFF-DIAGONAL =  1.75E-01 &  1.29E-03
FOR JTOT =  20 IBLOCK =  1 ENERGY( 1) = 1250.000    :  MAXIMUM DIAGONAL & OFF-DIAGONAL =  1.88E-01 &  1.95E-06
FOR JTOT =  20 IBLOCK =  2 ENERGY( 1) = 1250.000    :  MAXIMUM DIAGONAL & OFF-DIAGONAL =  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   1260.0000000000000    IS NEVER OPEN

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

 1      0.0000000000000
 2     180.0000000000000
 3     600.0000000000000

```

*** 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	3.722329E-02
1250.000000	10	10	20	3	1	6.542681E-07
1250.000000	10	10	20	1	2	8.697029E-03
1250.000000	10	10	20	2	2	3.00836
1250.000000	10	10	20	3	2	8.161628E-05
1250.000000	10	10	20	1	3	1.398009E-07
1250.000000	10	10	20	2	3	7.464053E-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) 2018 J. M. Hutson & C. R. Le Sueur
|
|           Version 2019.0 (beta)
|
|           This run used      xxxx cpu secs and
|           1279 of the allocated 1000000 words of storage
|
----- MOLSCAT ----- MOLSCAT ----- MOLSCAT ----- MOLSCAT ----- MOLSCAT ---

```

3.7.2 Higher print level

The following example (whose output will not be 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-H₂ colliding with ortho-H₂. It also uses a realistic potential provided by a VRTP routine.

Only sections of the output file that differ in important ways from section 3.7.1 will be 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 = 1,   STEPS  = 15.0,
  JTOTL  = 10,    JTOTU  = 12,
  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 ALGORITHMS FOR COUPLED EQUATIONS SPECIFIED BY IPROPS = 6 AND IPROPL = 1

COUPLED EQUATIONS WILL BE PROPAGATED OUTWARDS IN 2 SEGMENTS
+++++
SEGMENT 1 WILL BE PROPAGATED OUTWARDS

FROM RMIN CHOSEN USING IRMSET = 9 TO RMID = 2.00

COUPLED EQUATIONS SOLVED BY DIABATIC MODIFIED LOG DERIVATIVE METHOD OF MANOLOPOULOS

PROPAGATION STEP SIZE DETERMINED USING STEPS = 15.0 (PER WAVELENGTH)

LOG-DERIVATIVE MATRIX WILL BE INITIALISED IN THE LOCAL EIGENBASIS FOR RMIN IN THE OUTWARDS PROPAGATION PART
LOCALLY CLOSED CHANNELS WILL BE INITIALISED WITH A WKB BOUNDARY CONDITION
LOCALLY OPEN CHANNELS WILL BE INITIALISED WITH THE VALUE 1.0000000E+08
+++++
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 METHOD.
ALL PUBLICATIONS RESULTING FROM THE USE OF THIS PROPAGATOR SHOULD REFERENCE
M.H. ALEXANDER AND D.E. MANOLOPOULOS, J CHEM PHYS 86, 2044 (1987).

INITIAL STEP SIZE TAKEN FROM DR      = 2.000E-02
                                WITH TOLHI = 1.00E-04      POWRX = 3.0
TOLHI.LT.1  -- AIRY STEPS ADJUSTED TO MAINTAIN APPROX. ACCURACY VIA PERTURBATION THEORY AND POWRX.
+++++

```

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.

TARGET/PROJECTILE ENERGY LEVELS TAKEN FROM &BASIS (JLEVEL) INPUT.  NLEVEL = 3

ENERGY LEVELS COMPUTED FROM B(E) = 59.067000

PROJECTILE ENERGY LEVELS COMPUTED FROM B(E) = 59.067000

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

UNITS FOR POTENTIAL ROUTINES:
POTENTIAL RETURNED IN UNITS OF EPSIL = 24.170000 CM-1
R IN UNITS OF RM = 3.490000 ANGSTROMS

ALL LENGTHS ARE IN UNITS OF RM = 3.490000 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 *****

IBLOCK = 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 QUANTUM NUMBERS ---- L PAIR LEVEL PAIR ENERGY (CM-1)
              J1      J2      J12
1           2         0       3       4         2       708.8040000
2           4         2       1       4         3       472.5360000
3           5         2       1       4         3       472.5360000
4           1         0       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 8.1.2. This produces output as follows.

```

JTOT =    6  IBLOCK =    1  ENERGY(  1) =  700.00000000    CM-1
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:

```

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

AIRY.   DRMIN =    0.0407  DRMAX =  0.795
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 LIST:

OPEN CHANNEL	WVEC (1/ANG.)	CHANNEL	L	PAIR LEVEL	PAIR ENERGY (CM-1)
1	5.89835059E+00	4	6	1	118.1340000000000
2	3.68786851E+00	3	4	3	472.5360000000000
3	3.68786851E+00	2	4	3	472.5360000000000
4	3.68786851E+00	6	6	3	472.5360000000000
5	3.68786851E+00	7	6	3	472.5360000000000
6	3.68786851E+00	8	6	3	472.5360000000000
7	3.68786851E+00	10	8	3	472.5360000000000
8	3.68786851E+00	11	8	3	472.5360000000000

ROW	COL	S**2	PHASE/2PI	RE (S)	IM (S)
1	1	9.9281504753113E-001	-1.5519893420341E-001	5.5902994838289E-001	-8.2480334889121E-001
2	1	7.5823646190877E-004	1.4529408966605E-002	2.7421430002748E-002	2.5103064580200E-003

[ENTRIES FOR REMAINDER OF 8x8 S MATRIX]

7	8	1.2342644331010E-003	1.6556821897128E-003	3.5130198745915E-002	3.6547116640348E-004
8	8	9.9351391916290E-001	-2.4320444735069E-001	4.2546090531685E-002	-9.9584323532541E-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) FOR JTOT = 6 AND IBLOCK = 1 * * * * *
F I =      1      3
1      3.42292E-01  1.43838E-03
3      2.81148E-03  3.42742E+00

```

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) FOR JTOT = 6 AND IBLOCK = 2 * * * * *
F I =      1      3
1      1.94114E+00  2.07216E-03
3      4.05025E-03  2.96884E+00

* * * * *
FOR JTOT = 6 IBLOCK = 2 ENERGY( 1) = 700.0000 : MAXIMUM DIAGONAL & OFF-DIAGONAL = 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

* * * * *
FINISHED JTOT = 6 IBLOCK = 2 ENERGY( 1) = 700.0000

```

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

3.8 Example for BOUND

`bound-basic1.input` contains input data for a small calculation of the bound states for the model system in section 3.7.1. The only differences from the corresponding data 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 ALGORITHMS FOR COUPLED EQUATIONS SPECIFIED BY IPROPS = 6 AND IPROPL = 6
COUPLED EQUATIONS WILL BE PROPAGATED TOWARDS RMATCH IN 2 SEGMENTS
+++++

```

```
SEGMENT 1 WILL BE PROPAGATED OUTWARDS
```

```
FROM RMIN   =      0.50  TO RMATCH =      1.00
```

```
COUPLED EQUATIONS SOLVED BY DIABATIC MODIFIED LOG DERIVATIVE METHOD OF MANOLOPOULOS
```

```
PROPAGATION STEP SIZE DETERMINED USING DR = 1.000E-03
```

```
LOG-DERIVATIVE MATRIX WILL BE INITIALISED IN THE LOCAL EIGENBASIS FOR RMIN IN THE OUTWARDS PROPAGATION PART
```

```
LOCALLY CLOSED CHANNELS WILL BE INITIALISED WITH A WKB BOUNDARY CONDITION
```

```
LOCALLY OPEN CHANNELS WILL BE INITIALISED WITH THE VALUE 1.0000000E+30
```

```
+++++
SEGMENT 2 WILL BE PROPAGATED INWARDS
```

```
FROM RMAX   =     20.00  TO RMATCH =      1.00
```

```
COUPLED EQUATIONS SOLVED BY DIABATIC MODIFIED LOG DERIVATIVE METHOD OF MANOLOPOULOS
```

```
PROPAGATION STEP SIZE DETERMINED USING DR = 1.000E-03
```

```
Y MATRIX WILL BE INITIALISED IN THE LOCAL EIGENBASIS FOR RMAX IN THE INWARDS PROPAGATION PART
```

```
LOCALLY CLOSED CHANNELS WILL BE INITIALISED WITH A WKB BOUNDARY CONDITION
```

```
LOCALLY OPEN CHANNELS WILL BE INITIALISED WITH THE VALUE 0.0000000E+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 data 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 =====
```

```
NODE COUNT = 3 AT MINIMUM ENERGY = -10.00000000 CM-1
```

```
NODE COUNT = 5 AT MAXIMUM ENERGY = -1.000000000 CM-1
```

```
CONVERGED ON NODE NUMBER 4 AT ENERGY = -4.992074666 CM-1
```

```
CONVERGED ON NODE NUMBER 5 AT ENERGY = -1.413103889 CM-1
```


3.9 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.

3.9.1 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 $100 \text{ nK} \times k_B$, 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 data file that specifies this calculation is provided as `molscat-basic_Rb2.input`.

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

$$\text{JTOTL} = 8, \quad \text{JTOTU} = 8, \quad \text{IBFIX} = 2,$$

Setting **IBFIX** = 2 specifies that only the symmetry block with $j + L + J_{\text{tot}}$ 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 in temperature units (**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).

$$\text{EUNITS} = 2, \quad \text{NNRG} = 1, \quad \text{ENERGY} = 1.\text{E-}9, \quad \text{MONQN} = 4, 4, 4, 4,$$

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

$$\text{FLDMIN} = 800.0, \quad \text{FLDMAX} = 900.0, \quad \text{DFIELD} = 1.0,$$

for any channels that have low enough energy:

```

EFV SET      1:  MAGNETIC Z FIELD =   800.0000000      GAUSS

REFERENCE ENERGY EREF = -0.2561035762      K = -0.1780008389      CM-1

DWPROP. LOG DERIVATIVE MATRIX PROPAGATED FROM      3.0000 TO 21.000      IN 9000 STEPS.

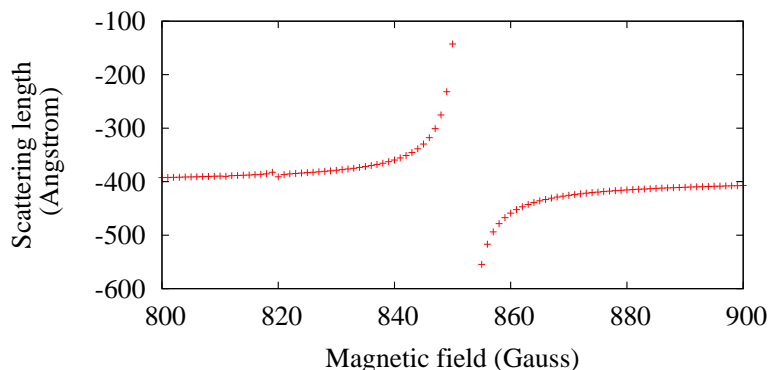
AIRPRP. LOG DERIVATIVE MATRIX PROPAGATED FROM      21.0000 TO 15000.      IN 1286 STEPS.

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  2.2139871055772E-004  -3.9108834844052E+002  -5.5463348015176E-006
  2    2  4  2.2139871055772E-004  -3.9961974519441E+009  -5.0726742501550E+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.

3.9.2 Using FIELD to identify 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 data 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 data file that specifies this calculation is provided as **field-basic_Rb2.input**. The resulting output identifying the fields at which zero-energy bound states exist includes

```
CONVERGED ON NODE NUMBER 1469 AT          MAGNETIC Z FIELD = 851.8811700    GAUSS
.
.
CONVERGED ON NODE NUMBER 1470 AT          MAGNETIC Z FIELD = 819.5674044    GAUSS
.
.
CONVERGED ON NODE NUMBER 1471 AT          MAGNETIC Z FIELD = 810.9519396    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.

3.9.3 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`,

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

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 $\Phi_i(\xi_{\text{intl}})$ 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 4.4, 4.5, 4.6 and 4.7.

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_{intl} of the separated monomers and \hat{L}^2 ;
2. for basis sets that are non-diagonal in H_{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 4.8. 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_{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 4.5. 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 selects the *pair basis functions* that will actually be used to solve each set of coupled equations. Each basis function is specified by an element of the array **JSINDEX** (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_{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 **JSINDEX**, 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_{intl} or \hat{L}^2

For the built-in interaction types, the basis functions are eigenfunctions of the Hamiltonian H_{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_{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_{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 2.5) in the primitive basis set. They use the channel threshold energies only for reference energies. However, MOLSCAT transforms the wavefunction matrix $\Psi(R_{\text{max}})$ (or the log-derivative matrix $\mathbf{Y}(R_{\text{max}})$) into the *asymptotic basis set* that diagonalises H_{intl} and \hat{L}^2 (and, optionally, extra operators to resolve degeneracies in the eigenvalues of H_{intl}). Scattering boundary conditions are then applied in the asymptotic basis set.

4.3 Units of energy

EUNITS: an integer specifying the energy units for ELEVEL and spectroscopic constants input in &BASIS. EUNITS is distinct from the EUNITS parameter of namelist &INPUT (section 6.1), but its allowed values and their meanings are the same.

4.4 Specification of 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 will be 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 data 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.

4.4.1 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

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

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

4.4.2 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 $j_1, v_1, j_2, v_2, \dots, j_{\text{NLEVEL}}, v_{\text{NLEVEL}}$). 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

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

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.

4.4.3 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 $j_{11}, j_{21}; j_{12}, j_{22}; \dots; j_{1, \text{NLEVEL}}, j_{2, \text{NLEVEL}}$).

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. It is also necessary in this case to specify the statistical weights to be applied to the different symmetry combinations when calculating cross sections. The statistical weights for anti-symmetric and symmetric combinations of (j_1, j_2) and (j_2, j_1) may either be specified explicitly in the **WT** array as **WT(1)** and **WT(2)** respectively, or are calculated from the single nuclear spin input in **SPNUC** if **WT** is not specified explicitly. The weight information does not affect the results of bound-state calculations, but none of the programs will perform a calculation for any symmetry block with a statistical weight of zero.

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

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

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 $\text{BE} - 0.5 * \text{ALPHAE}$, these values may be input directly in **BE(1,2)** with **ALPHAE** omitted from the namelist if preferred.

4.4.4 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_{mk}^{j*} , 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, PRTY) . **PRTY** indicates that the function is proportional to an even or odd linear combination of rotation matrices,

$$D_{mk}^{j*} + (-1)^{\text{PRTY}} D_{m-k}^{j*}, \quad (4.3)$$

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

The array **JLEVEL** is usually generated from input parameters **JMIN**, **JMAX** and **JSTEP**. If **JSTEP** = 2, only functions of the same parity as $j = \text{JMIN}$, $k = 0$ 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 $k = |\text{KSET}|$ are included; if it is zero or positive, only levels with $k \leq \text{KMAX}$ 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, 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

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

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 ,

$$\nu = \nu_0 - \nu_a [j(j + 1) - k^2] - \nu_b k^2. \quad (4.5)$$

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 $D_{m\Lambda}^{j*}$ or D_{ml}^{j*} . In this case the basis set will usually contain 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 CO₂ or HCCH), JSTEP should be set to 2 to include only functions of the same parity as $j = \text{JMIN}$, $k = 0$ as above.

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

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

$$|j, \tau, m\rangle = \left(\frac{2j+1}{8\pi^2} \right)^{\frac{1}{2}} \sum_k a_{\tau,k}^j D_{mk}^{j*}. \quad (4.6)$$

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, $1 \leq \tau \leq 2j+1$. JLEVEL is usually generated from input parameters JMIN, JMAX and JSTEP as described below.

The monomer energy levels $E_{j\tau}$ and wavefunction coefficients $a_{\tau,k}^j$ 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

$$\hat{H}_{\text{rot}} = A j_a^2 + B j_b^2 + C j_c^2. \quad (4.7)$$

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

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

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 if the corresponding wavefunction coefficients $a_{\tau,k}^j$ 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 **EUNITS**, so that subsequent calculations must use the same value for **EUNITS**.

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 will not be 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 $a_{\tau,k}^j$. Each level is described on **IASYMU** in the format $j, \tau, E_{j\tau}$ (2I5,F15.10) where $E_{j\tau}$ is the rotor energy in **&BASIS** energy units. For each level, $2j + 1$ wavefunction coefficients $a_{\tau,k}^j$ are required (corresponding to $k = -j, -j + 1, \dots, j$); 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 $E_{j\tau} > \text{EMAX}$ (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**($2i$), $i = 1, |\text{NLEVEL}|$.

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 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 $\sum_k c_k (D_{mk}^{j*} + \epsilon D_{m,-k}^{j*})$ 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.

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 $J_{\text{tot}} = 0$ 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 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.

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

The basis functions here are combinations of asymmetric top rotor wavefunctions and linear rotor wavefunctions [22]. 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, **ALPHA**E(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 will use **|NLEVEL|** sets of quantum numbers from the array **JLEVEL** and will 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 4.4.5.

4.4.7 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 [27, 28] 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 \mathbf{k} may be decomposed into components $k_\perp = k \cos \theta$, perpendicular to the surface, and a vector \mathbf{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 wave vector 100 Å⁻¹.

..
..

ENDDO
ENDDO

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

$$\exp[i(\mathbf{K} + \mathbf{G}) \cdot \mathbf{R}], \quad (4.9)$$

where \mathbf{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 \mathbf{K} of the momentum enters in the threshold energies $\hbar^2 |\mathbf{K} + \mathbf{G}|^2 / 2m$. 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 \mathbf{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 $\mathbf{G} = (g_1, g_2)$ is included only if the parallel kinetic energy $\hbar^2 |\mathbf{G}|^2 / 2m < EMAX$.

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 \mathbf{K} of the incident momentum, and selects from the master list those basis functions for which $\hbar^2|\mathbf{K}+\mathbf{G}|^2/2m < \text{EMAXK}$. 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.

4.5 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 diagonal basis sets. 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 4.4.5. 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 $a_{\tau,k}^j$ 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.

4.6 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_{tot}

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

The resulting coupled equations are independent of M_{tot} . The calculations are carried out for one value of J_{tot} and symmetry block `IBLOCK` at a time. For close-coupling calculations, `IBLOCK` encodes the total parity $(-1)^{j+L}$, which is $(-1)^{J_{\text{tot}}+\text{IBLOCK}}$. 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_{tot} and total parity. Inside the loops over `JTOT` and `IBLOCK`, the programs construct arrays `JSINDEX` and `L`, of dimension N . For each basis function i , `JSINDEX(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_{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)`.

4.7 Decoupling approximations

4.7.1 Effective potential

The effective potential method of Rabitz [29] 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_{tot} are identical.

4.7.2 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 $K = \text{IBLOCK} - 1$. 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

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

This is invoked with `IBOUND = 1`. In this approach `JTOT` is interpreted as J_{tot} , so $K \leq J_{\text{tot}}$ 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.¹

The L -labelled coupled-states approximation of McGuire and Kouri [30] 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_{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_{tot} . In this case, although the basis set is still restricted to functions $j \geq K$, K is *not* restricted by $K \leq \text{JTOT}$.

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 $j > \text{JZCSMX}$ 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 $L(i) = \text{IABS}(\text{JTOT} + \text{JZCSFL} * j(i))$. This is a historical remnant and values other than the default are not recommended.

4.7.3 Decoupled L -Dominant

This decoupling approximation [32] 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.

4.8 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 programs perform calculations only for `IBLOCK = IBFIX`. If `IBHI` \geq `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

¹In versions of MOLSCAT before 2014, the integer values contained in `L` were used in the scattering boundary conditions.

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).

4.9 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.

4.10 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 [44, 24] 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 4.4.3), `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` \neq 0 requests Gauss-Legendre.

4.11 The BCT Hamiltonian

The Bohn-Cavagnero-Ticknor (BCT) Hamiltonian [45] 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 the “linear molecule” rotational quantum number is interpreted as 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.

4.12 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_cp1d.f` handles interactions of a structureless (1S) atom and a molecule in a $^3\Sigma$ state in a magnetic field;

`base9-alk.alk.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 variable input in namelist `&BASIS`, which are placed in module `efvs` 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 \mathbf{R} .

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

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

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 <i>must</i> 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 coeffi-

*This is not true for the IOS code in MOLSCAT, which can use unexpanded potentials directly; see section 5.5.

coefficients $v_\lambda(R)$, are provided by subroutine POTENL. Previous versions of the programs 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

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

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. 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

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

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 [19],

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

The interaction potential actually depends only on the relative angle $\phi_1 - \phi_2$. The spherical harmonics Y_{λ}^{μ} 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,

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

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 [23],[§]

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

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

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

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 **MMA** 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 $\lambda, |\kappa|$.

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 [22]. This is a generalisation of the expansion for ITYP = 3, with an extra angle χ representing the rotation of the asymmetric

[‡] **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,

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

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 \mathbf{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:

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

where \mathbf{r}_j is the j th lattice site. This gives for the Fourier components of the atom-surface potential

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

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 distance and energy used by the potential routine. They are specified in namelist **&POTL** in units of Ångstrom and cm^{-1} respectively. **RM** is used as the unit of length throughout the remainder of the programs, and many quantities are printed in “RM units”.

Values of **RM** and **EPSIL** set in the 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.

5.3 Evaluating the radial potential coefficients

5.3.1 Potential supplied by VINIT/VSTAR 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`, POTENL obtains the coefficient by a call to `VSTAR`, which is specified below.
- If `NTERM(i) ≥ 0`, the i th 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`, which are interpreted as in the code fragment:

```

IEXP = 0
ITERM = 0
DO ICOEFF = 1, MXLAM
  P(ICOEFF) = 0.DO
  IF (NTERM(I).LT.0) CALL VSTAR(R, ICOEFF, P(ICOEFF))
  DO JTERM = 1, NTERM(I)
    ITERM = ITERM + 1
    IF (NPOWER(ITERM).LT.0) THEN
      P(ICOEFF) = P(ICOEFF) + A(ITERM) * R**NPOWER(ITERM)
    ELSEIF (NPOWER(ITERM).EQ.0) THEN
      IEXP = IEXP + 1
      P(ICOEFF) = P(ICOEFF) + 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 data file would contain: `NTERM` = -1, 2, 2, `NPOWER` = -12, -6, 0, -6, `A` = C_{12} , $-C_6$, a , $-b$ and `E` = $-\beta$.

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`. `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 `RMUNIT` and `EPUNIT` in module `potential`. These variables are pre-populated with 'RM UNITS' and 'EPSIL UNITS' respectively, but if `RM` = 1.0 after the call to `VINIT`, the general-purpose potential routine sets `RMUNIT` to 'ANGSTROM'; similarly, if `EPSIL` = 1.0, it sets `EPUNIT` 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.

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 $V(R, \xi_{\text{intl}})$ explicitly as functions of internal coordinates ξ_{intl} , rather than as an expansion over internal functions. In such cases `LV RTP` 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

$$v_{\Lambda}(R) = N_{\lambda} \int d\xi_{\text{intl}} V(R, \xi_{\text{intl}}) \mathbf{v}^{\Lambda}(\xi_{\text{intl}}), \quad (5.11)$$

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

$$v_{\Lambda}(R) = N_{\lambda} \sum_{i=1}^n w_i \mathbf{v}^{\Lambda}(\xi_i) V(R, \xi_i), \quad (5.12)$$

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

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	i th root of $P_n(x)$	$2(P_n(1))^2 / [(1 - x_i^2)(P'_n(x_i))^2]$
Gauss-Hermite	i th 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 $\theta \geq 90^\circ$ are used. If ICNSYM > 1, NPTS(2) points are used in the range $0 < \phi_1 - \phi_2 < \pi/\text{ICNSYM}$.

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

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

```
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 distance and energy to be used in subsequent calls by overwriting the energy and length units factors `RM` (= `R`) and `EPSIL` (= `V`). It may also set the variables `IHOMO`, `IHOMO2`, `ICNSYM`, and `ICNSY2` in module `angles`; see section 5.4 for specification.[¶]

If desired, an initialisation call to `VRTP` may supply names for the units `RM` and `EPSIL` in the character variables `RMUNIT` and `EPUNIT` 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 must be given in the `COSANG` array. `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 `NUMBER=.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`).

5.4 Module angles

The specification of module `angles` is

```
USE sizes, ONLY: MXANG
```

```
INTEGER                IHOMO, ICNSYM, IHOMO2, ICNSY2
```

```
DOUBLE PRECISION COSANG(MXANG), FACTOR
```

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

[¶]The data included in module `angles` were previously in a common block named `ANGLES`. Earlier versions of `VRTP` must be modified to use module `angles`.

5.5 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 data 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 Energy units

The units used for all energies input in namelist &INPUT are specified by the integer variable EUNITS as follows:

EUNITS = 1 cm^{-1}	EUNITS = 4 GHz	EUNITS = 7 Hartrees
EUNITS = 2 Kelvin	EUNITS = 5 eV	EUNITS = 8 kJ/mol
EUNITS = 3 MHz	EUNITS = 6 erg	EUNITS = 9 kcal/mol

The input quantities affected by EUNITS are ENERGY, DNRG, EMIN and EMAX, EREF; and 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** will calculate 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 **MONQN**(1) = -99999 (the default) and **IREF** > 0 , the behaviour depends on whether H_{intl} is diagonal or non-diagonal in the basis set:

Diagonal H_{intl} : The programs use the energy of the pair level with index **IREF**.

Non-diagonal H_{intl} : The programs use the asymptotic energy of the channel with index **IREF** as the zero of energy. 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) $\neq -99999$ and **ITYP** $\neq 9$, the values supplied in the array **MONQN** are quantum labels for the reference threshold, which must match those in a row of the **JLEVEL** array).
- If **MONQN**(1) $\neq -99999$ and **ITYP** = 9, 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_{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_{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_{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

In version 2007.1, MOLSCAT and BOUND were extended to incorporate the effects of external magnetic and/or electric fields. Version 2019.0 (beta) introduces 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 **MAXEFV** = 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** will mean that the grid of values held in the **FIELD** array will 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

Previous versions of the programs have implemented a variety of propagators to solve the coupled equations, and have 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. In 2019.0 (beta), this mechanism has been 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** and **IPROPL**. If only **IPROPS** has been specified then **IPROPL** is assumed to be the same as **IPROPS**. If neither are specified, the default combinations are **IPROPS** = **IPROPL** = 6 for BOUND and FIELD, and **IPROPS** = 6 and **IPROPL** = 1 for MOLSCAT.

The propagator codes are:

- | | |
|---|--|
| 1 Alexander-Manolopoulos Airy (AIRY) | 7 Manolopoulos quasi-adiabatic modified log-derivative (LDMA) |
| 2 de Vogelaere (DV) | |
| 3 R-matrix (RMAT) | 9 Manolopoulos-Gray symplectic log-derivative (LDMG) |
| 5 Johnson log-derivative (LDJ) | |
| 6 Manolopoulos diabatic modified log-derivative (LDMD) | 14 VIVS |
| | -1 WKB phase integrals by quadrature |

If **IPROPL** = 0, it is set the same as **IPROPS**. If **RMID** is not between **RMIN** and **RMAX**, **IPROPL** is not used.

The AIRY and VIVS propagators are not currently allowed at short range.

The DV and RMAT propagators may not currently be used if H_{intl} or \hat{L}^2 is non-diagonal.

For backwards compatibility, **INTFLG** is a deprecated synonym for **IPROPS**. There are two additional values of **INTFLG** that have special meanings:

INTFLG = 4 sets **IPROPS** = 5 and **IPROPL** = 14, corresponding to the VIVAS hybrid propagator of Parker *et al.* [42].

INTFLG = 8 sets **IPROPS** = 6 and **IPROPL** = 1, corresponding to the hybrid log-derivative/Airy propagator of Alexander and Monolopoulos [35, 36].

8.1.1 Units of length

The programs operate with units of lengths specified by **RM** (chapter 16), which may be input via namelist &POTL, as described in section 11.3 (but note that this value may be over-written by a user-supplied potential routine VINIT or VRTP). Input length variables that control the integration (**RMIN**, **RMAX**, **RMID**, **RMATCH**, **DR**) are in units of **RM**.

RM itself is specified in units of Å. For example **RM** = 1.0 specifies that all distances are in Å, while **RM** = 0.52917721067 (2014 value) specifies that all distances are in bohr radii (atomic units). Some output quantities (such as scattering lengths, but not cross sections) are output in these “RM units”.

8.1.2 Ranges of integration

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.

Inner limit R_{\min}

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 $10^{-\text{IRMSET}}$ 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).

Outer limit R_{\max}

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

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

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

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. How-

ever, it can be drastically inefficient (and is unnecessary) for ultracold scattering including channels with $L > 0$, so the default value of **IRXSET** been changed from 1 to 0 in version 2019.0 (beta).

The de Vogelaere and R-matrix propagators also test some aspects of convergence internally, as described below, and may propagate beyond R_{\max} to achieve convergence under certain circumstances.

Propagator switch point R_{mid}

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_{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 1.1 to 1.4 are often satisfactory for cross-section calculations.

RVIVAS is a deprecated synonym for **RMID**.

Bound-state matching point R_{match}

R_{match} is set to **RMATCH**. If no value for **RMATCH** is provided, R_{match} is set to **RMID**.

The value of **RMATCH** does not affect converged bound-state energies or fields, but it can affect the rate at which the programs converge to them (and sometimes the success of convergence). A value of **RMATCH** 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.

8.1.3 Step size

The de Vogelaere and the LDJ propagators are wavefunction-following methods, while all the remaining propagators are potential-following methods.

The default behaviour (**IABSDR** = 1) is to take the step size δr from **DR** for the short-range part of a propagation and **DRLR** for the long-range part. If **DRLR** is left unset it is set internally to be the same as **DR** for outwards propagations and $R_{\max}/400$ for inwards propagations (with the single exception of the Airy propagator with **TOLHI** > 1; see section 8.3.6 for more details).

- For the LDJ, LDMD, LDMA and LDMG propagators, δr is constant throughout the propagation (the value given in either **DR** or **DRLR** may be modified slightly if necessary to give an integer number of steps over the propagation range).
- For the R-matrix propagator, **DR** is used to set a constant value for δr which is used from R_{\min} to R_{mid} , but then δr is increased outside R_{mid} as described below (**DRLR** is not used in this case, and this propagator cannot be combined with another propagator for the long-range part).

- For the AIRY and VIVS propagators, **DRLR** is used to set the *initial* value of δr . However, both the AIRY and VIVS propagators adjust the step size internally and may use step sizes much larger than **DRLR** in regions where the potential varies slowly.

If **IABSDR** = 0 and **STEPS** > 0.0, the step size is calculated from **STEPS**. This is interpreted as the number of steps per half-wavelength in the asymptotic region for the open channel of highest kinetic energy. A value between 10 and 20 is usually adequate, unless the depth of the potential well is large compared to the kinetic energy, in which case using the **DR** mechanism is recommended.

The two modified log-derivative propagators are actually potential-following methods, but they may nevertheless use the **STEPS** mechanism to determine step size, for consistency with the basic log-derivative code.

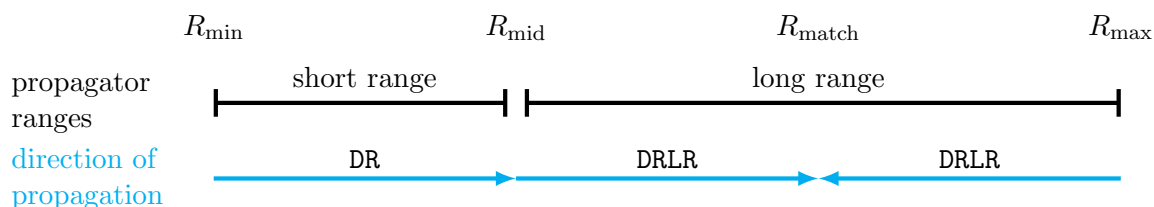
If **ISCRU** > 0 and **STEPS** > 0, **DR** is calculated from **STEPS** at the highest energy in **ENERGY** and this value is used for all the fixed-step-size propagations.

8.1.4 3-segment propagation in **BOUND** and **FIELD**

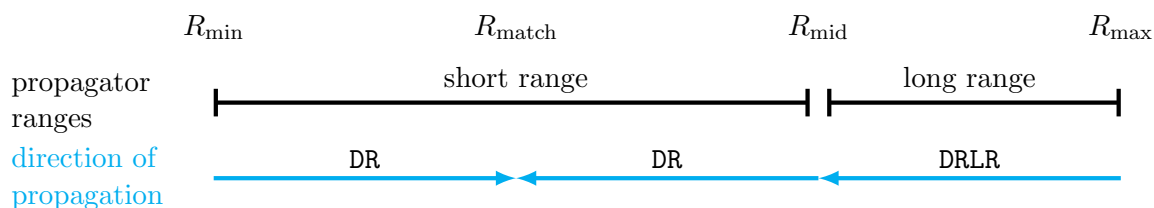
The wavefunction matching distance **RMATCH** may be different from the propagator switching point **RMID**. This is a new facility in version 2019.0 (beta); for backwards compatibility, if either of **RMID** or **RMATCH** is omitted, it is set internally to be the same as the other value. If both **RMID** and **RMATCH** are set and are not equal to each other, 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: $R_{\text{mid}} < R_{\text{match}}$



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



8.2 Convergence of propagations

It is very important to test convergence of coupled-channel calculations with respect to integration 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.

8.3 Propagators in more detail

8.3.1 de Vogelaere propagator (propagator 02)

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

STEST is an integration tolerance parameter used for two purposes:

1. The K matrix obtained by matching to the asymptotic boundary conditions should be symmetric, but this is never exactly the case because of numerical rounding errors. The program counts and prints out the number of pairs of symmetry-related off-diagonal elements that differ by more than **STEST**.
2. The S matrix is calculated at successive values of R , starting at **RMAX** and incrementing by approximately half a wavelength each time. This is repeated until two successive S matrices differ by no more than **STEST** in any element, up to a maximum of 20 attempts.

8.3.2 R-matrix propagator (propagator 03)

For $R < R_{\text{mid}}$, the R-matrix propagator uses a constant step size of **DR**. For $R > R_{\text{mid}}$, the step size is $\text{DR} \times R/R_{\text{mid}}$, so that larger steps are taken in the long-range region.

SHRINK If the integer variable **SHRINK** is 1 on input, the R-matrix propagator monitors closed channels in the long-range region and accumulates a phase integral for each. If a semi-classical estimate based on this phase integral indicates that the wavefunction amplitude in the channel concerned has dropped below $10^{-\text{IRMSET}}$, the channel is removed from the basis set for the remainder of the propagation.

VTOL controls a convergence criterion applied to the eigenvectors of the potential matrix. The R-matrix propagator does not attempt to apply boundary conditions until the eigenvector is a (permuted) unit matrix to within tolerance **VTOL** for two successive steps. This test is applied only to eigenvectors that are asymptotically non-degenerate.

MAXSTP If convergence has not been achieved after this many steps, the program prints an error message and exits.

8.3.3 VIVS propagator (propagator 14)

Control of the VIVS propagator is quite complicated. VIVS operates 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 algorithms used to determine interval and step sizes are discussed separately.

The step size and interval size algorithms used by VIVS attempt to take the longest step that will give the required accuracy at each point in the integration. However, the optimum step size at one scattering energy is not necessarily safe at another, and VIVS can sometimes give problems 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.

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.

8.3.4 LDJ, LDMD and LDMA propagators (propagators 05, 06 and 07)

The input variables for the propagation end points and step length are as described in sections 8.1.2 and 8.1.3.

The LDMA propagator has special uses because it works in a locally adiabatic basis. It therefore provides a convenient way to print adiabats (eigenvalues of $H_{\text{intl}} + V(R, \xi_{\text{intl}}) +$

$\hbar^2 \hat{L}^2 / (2\mu R^2)$ at fixed values of R), which are often useful in interpreting bound states and scattering, as described in section 12.5.

8.3.5 LDMG propagator (propagator 9)

The input variables for the propagation end points and step length are as described in sections 8.1.2 and 8.1.3. 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).

8.3.6 AIRY propagator (propagator 01)

The AIRY propagator is available only at long range, between R_{mid} and R_{max} . It was originally developed for use in combination with the LDMD propagator at short range, but in version 2019.0 (beta) of the programs it can be combined with any other log-derivative propagator. (Not all combinations are recommended, but the flexibility may be useful for advanced users.)

By default the initial step size for continuing outward propagations is taken as the value calculated for the short-range part, but it can be further controlled by the following variables:

DRLR is the initial step size, except in the special case **TOLHI** > 1 (see below). In this case **DRLR** sets the size of the smallest step in the geometric series regardless of the direction of propagation.

If a 3-segment propagation is being performed as in case 1 of section 8.1.4, **DRLR** should be set to a value suitable for the outwards propagation from R_{mid} ; if **TOLHI** < 1.0, the inwards propagation will quickly adjust to an appropriate value for the step size. Alternatively, leaving **DRLR** at its default value will result in a sensible choice for the initial step size for both segments.

TOLHI In normal circumstances **TOLHI** is set to a small value; it is then used as a tolerance to adjust the step size (by a perturbation method) to try to maintain accuracy, as described in ref. [35]. Values between 10^{-3} and 10^{-6} are generally useful. In special circumstances, **TOLHI** can instead be set to a value greater than 1, in which case the step size is scaled by this factor for each step (step size is multiplied by **TOLHI** for outwards propagations, and divided by it for inwards propagations). Note that, for outwards propagations, a final step is taken from the last point in the geometric series that is within **RMAX** or **RMATCH**, as appropriate, to the endpoint of the propagation. This step is always smaller than **TOLHI** × the previous step. Similarly, for inwards propagations, an initial step is taken from **RMAX** to the beginning of the geometrically decreasing series; this is always smaller than **TOLHI** × the next step.

POWRX (default 3.0/6.0) The step-size algorithm used for **TOLHI** < 1.0 is based on the expectation that the radial potential coefficients decrease (or increase for inward propagation) proportional to $R^{-\text{POWRX}}$. The default is usually adequate for outwards propagations, but **TOLHI** sometimes needs to be increased for inwards propagations to prevent over-long steps at short range.

The Airy propagator uses a linear reference potential $W(R) = W_0 + W_1(R - R_0)$ in each step. The independent solutions of this piecewise linear potential are Airy functions of argument $W_0 W_1^{-2/3} + W_1^{1/3}(R - R_0)$. For small arguments, the Airy functions are evaluated using the routines of Alexander and Manolopoulos [36]. However, as W_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. In version 2019.0 (beta), we have implemented a modified algorithm for evaluating the Airy functions which uses the expansions given in appendix A of ref. [46], which remains accurate as $W_1 \rightarrow 0$. The asymptotic expansions are used if $\log(|W_0 W_1^{-2/3}|) > \text{AC}$, the value of which is hard-coded in subroutine `SPROPN` but can be changed for special purposes.

8.3.7 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 `TOL`.

NGMP Dimension 3. N -point Gaussian integration is performed starting with $N = \text{NGMP}(1)$, 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 `TOL`.

TOL is the convergence tolerance for the WKB phase shift.

8.4 Boundary conditions for log-derivative propagators

All the programs carry out outwards propagations starting from R_{\min} , and `BOUND` and `FIELD` also carry out inward propagations starting at R_{\max} . Boundary conditions are needed to initialise these propagations. Previous versions of the programs have used a variety of boundary conditions, which were adequate for most cases, but different values are needed for special purposes. In version 2019.0 (beta), the choice of initial 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.

For a single channel, the log-derivative is defined as $Y = \psi'(R)/\psi(R)$. At the wall of a box, $\psi(R_{\text{wall}}) = 0$, 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

$$\psi(R) = [k(R)]^{-\frac{1}{2}} \exp\left(\pm \int_{R_{\text{turn}}}^R k(R') dR'\right), \quad (8.1)$$

$$\psi'(R) = [k(R)]^{-\frac{1}{2}} \left[\pm k(R) - \frac{1}{2} \frac{k'(R)}{k(R)} \right] \exp\left(\pm \int_{R_{\text{turn}}}^R k(R') dR'\right), \quad (8.2)$$

$$Y(R) = \pm k(R) - \frac{1}{2} \frac{k'(R)}{k(R)}, \quad (8.3)$$

where $k(R) = [2\mu(E - V(R))/\hbar^2]^{1/2}$ and $V(R)$ is an effective potential energy for the channel concerned. The first term in Eq. 8.3 dominates either when $k(R)$ is large (in a strongly classically forbidden region) or 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.

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 $\mathbf{W}(R)$. In previous versions of the programs the boundary conditions were applied in the primitive basis set for diabatic propagators and in the adiabatic basis set for adiabatic propagators. In version 2019.0 (beta) 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_{min} and R_{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.

The variables **BCYCMN**, **BCYCMX**, **BCYOMN** and **BCYOMX** specify the values to be used at R_{min} and R_{max} for closed (**BCYC**...) and open (**BCYO**...) channels. For closed channels, if **WKBMN** or **WKBMX** (as appropriate) is `.TRUE.` then the WKB boundary condition of Eq. 8.3 is used. Only the variables ...MN are used in MOLSCAT.

The default is to use WKB boundary conditions for closed channels (**WKBMN** = **WKBMX** = `.TRUE.`), with **BCYOMN** = 10^8 for open channels at R_{min} (corresponding to a hard wall) and **BCYOMX** = 0.0 for open channels at R_{max} . The reason for the latter choice is that it provides continuity across a threshold when WKB boundary conditions are used for closed channels, since Eq. 8.3 gives $Y(R_{\text{max}}) \rightarrow 0$ as a bound state approaches a threshold from below.

It is relatively rare to start a propagation in the presence of open channels. However, it is required for special purposes. For systems that have no classically forbidden region at short range, it may be appropriate to start the propagation at $R = 0$ and use finite $Y(0)$ for open channels with $L = 0$. It is also sometimes desired to use **BOUND** or **FIELD** to estimate the positions of quasibound states that lie above a dissociation threshold, and to do this the open channels must be quantised by applying a boundary condition at long range.

If open channels are present and are initialised with very large values (usually as a proxy for ∞), the initialisation should be done in the same representation (diabatic or adiabatic) as the propagation. Otherwise, there is a danger that the basis-set transformation will cause the open channels to contaminate the rest of the matrix in an unphysical way. For this reason a warning is issued if open channels are present at R_{min} or R_{max} .

8.5 Internal units of length and energy

The programs operate internally with lengths in units of `RM`, set in namelist `&POTL` or returned by a potential routine `VINIT` or `VRTP`. Quantities with units of energy E (total energy, potential energy, interaction matrix) are stored internally as $2\mu E/\hbar^2$ in units of `RM`⁻². The programs use two conversion factors: dividing by `CINT` converts energies in internal units into wavenumbers in cm^{-1} , while dividing by `RMLMDA` = `CINT` \times `EPSIL` converts energies in internal units into the units `EPSIL` used for the interaction potential.

8.6 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, R-matrix and VIVS propagators if `NNRG` is not 1, but for the de Vogelaere 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 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 $\Psi(R_{\text{max}})$ and its derivative $\Psi'(R_{\text{max}})$, or the log-derivative matrix $\mathbf{Y}(R_{\text{max}})$, 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 2.4. 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 \mathbf{H}_{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_{intl} and the extra operators. It is treated as an energy and so is scaled by **EUNITS**.

9.3 Channel indices for open channels

The S matrix has dimension $N_{\text{open}} \times N_{\text{open}}$, where N_{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. 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.

9.5 Automated convergence testing

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 integration 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 or external field variables; thus **JTOTU** must be equal to **JTOTL**, **IBFIX** must be greater than **IBHI** and **NFIELD** must be 1.

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 $|\text{ICONVU}|$, 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_{intl} and \hat{L}^2 are diagonal.

If one or both of H_{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_{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_{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 will calculate 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 will be calculated. If the reference energy is obtained from **IREF**, cross sections will only be calculated in runs that are limited to a single value of **JTOT** and a single symmetry block.

*MOLSCAT will, 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 cross section calculations

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 `JTOTLU` in steps of `JSTEP`. However, if `JTOTU` ≥ 99999 (or `JTOTU` $< \text{JTOTL}$) (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 than `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 will converge 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, $n_a \rightarrow n_b$ and $n'_a \rightarrow n'_b$. The real and imaginary parts of diagonal cross sections ($n_a = n'_a$ and $n_b = n'_b$) 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 [18]

$$\sigma(v_a j_a, v_b j_b | v'_a j'_a, v'_b j'_b; E_{\text{kin}}) = \frac{\pi}{k_j^2} \sum_{\substack{LL' \\ J_a J_b}} \left\{ \begin{matrix} j_a & q & j_b \\ J_b & L & J_a \end{matrix} \right\} \left\{ \begin{matrix} j'_a & q & j'_b \\ J_b & L' & J_a \end{matrix} \right\} \quad (9.1)$$

$$\times \left[I - S_{i_b f_b}^{J_b}(E_b^{\text{kin}})^* S_{i_a f_a}^{J_b}(E_a^{\text{kin}}) \right],$$

where $E_x^{\text{kin}} = E_{\text{kin}} + E_x$ and I is shorthand for $\delta_{v_a v'_a} \delta_{v_b v'_b} \delta_{j_a j'_a} \delta_{j_b j'_b} \delta_{L, L'}$. 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	[47]	[48]	[49]	[50]	[51]
vibrating rotor + atom	2	[47, 52]	✓	✓		
rigid rotor + rigid rotor	3	[53]				
asymmetric top + rigid rotor	4					
symmetric top + atom	5	[23]	✓	[23]		[54]
asymmetric top + atom	6	[23]	✓	[23]		
vibrating rotor + atom	7	[47, 26]	✓	✓		
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 $4 \times \text{NLPRBR}$ 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_{kin} in Eq. 9.1), and generates the necessary total energies for the lines requested (i.e., E_a^{kin} and E_b^{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 **JTOT**, **IBLOCK** (“symmetry block”) 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 Calculating line-shape cross sections within 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 applied 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.

This option is 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 [55], follows a Breit-Wigner form in the vicinity of a resonance

$$\mathcal{S}(E) = \mathcal{S}_{\text{bg}}(E) + \arctan\left(\frac{\Gamma}{2(E_{\text{res}} - E)}\right), \quad (9.2)$$

where E_{res} is the energy of the resonance, Γ is its width, and $\mathcal{S}_{\text{bg}}(E)$ is a slowly varying background phase. For full characterisation of resonances, MOLSCAT can output K matrices instead of S matrices on channel **ISAVEU**. The K-matrix files are 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.

In **MOLSCAT**, a run to save K matrices instead of S matrices is specified by setting **IPHSUM** > 0. Unit **IPHSUM** is used to write a summary of the eigenphase sums and scattering lengths obtained. See also section 13.5.1.

A **MOLSCAT** run to save K matrices for **SAVER** must deal with only one total angular momentum **JTOT** and symmetry block **IBLOCK** in a given run. Thus, **JTOTL** and **JTOTU** must be the same, and **IBFIX** must be set greater than 0 to select the particular symmetry block required.

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}/π 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}/π , 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 converge towards it when the curvature of the eigenphase sum is dominated by the resonant contribution. If **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 $|\mathbf{NNRG}|/5$. 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 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 desirable 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 8.6.

9.10 Calculating 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 2.4.1. These are output in units RM^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 0.2 RM^{-1} . This rather arbitrary threshold value may be changed by altering the value of the parameter **AWVMAX** in subroutine **OUTPUT**.

9.10.2 Scanning the scattering length/volume over external field

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 an external field

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). Once it has calculated scattering lengths/volumes at 3 different values (**FLDMIN**, **FLDMAX**, and $(\text{FLDMIN} + \text{FLDMAX})/2$), it uses the algorithms of Frye and Hutson [13] to converge on and characterise a 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,

$$a(B) = a_{\text{bg}} \left(1 - \frac{\Delta}{B - B_{\text{res}}} \right), \quad (9.3)$$

where B represents the varying EFV. In this case 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 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 parameters obtained are B_{res} , Δ and the real and imaginary parts of a_{res} and a_{bg} .

For any positive value of **IFCONV**, **TOLMIN** and **TOLMAX** set values for the parameters t_{\min} and t_{\max} of ref. [13]. The characterisation terminates when either the position of consecutive estimates of B_{res} are within **DTOL** of each other or when **MAXSCH** (set in module **sizes**) propagations have been performed without convergence. The current estimates of the resonance parameters are printed at each step.

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 field where the scattering length in open channel **ICHAN** satisfies $a - \text{AZERO} = 0$. This can be used to converge on a zero crossing if **AZERO** = 0.0. Convergence is attempted only if the value of $a - \text{AZERO}$ changes sign between **FLDMIN** and **FLDMAX**. Convergence uses the Van Wijngaarden-Dekker-Brent method [15] 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_{kin} or wavevector k is often characterised by an effective-range expansion at small collision momentum

$$k \cot \eta(k) = -\frac{1}{a_0(0)} + \frac{1}{2}r_{\text{eff}}k^2 + \dots \quad (9.4)$$

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} :

$$[a_0(k)]^{-1} = [a_0(0)]^{-1} - \frac{1}{2}r_{\text{eff}}k^2 + \dots, \quad (9.5)$$

or

$$a_0(k) = a_0(0) + \frac{1}{2}r_{\text{eff}}[a_0(0)]^2k^2 + \dots. \quad (9.6)$$

At applied 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. 9.5 is numerically unstable near a zero crossing and Eq. 9.6 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 always calculated for the second and subsequent energies, provided the wavevector for channel **ICHAN** is less than 0.2 RM^{-1} .

9.11 Calculating scattering wavefunction

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**. This feature is still under development, and its capabilities and input specifications may change in future, so will not be documented in detail here.

Chapter 10

Controlling bound-state calculations

10.1 The node count

The multichannel node count used by BOUND and FIELD was introduced by Johnson [14]. 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 (beta), the algorithm used required an additional propagation from either R_{\min} or R_{\max} to R_{mid} . In version 2019.0 (beta), this additional propagation has been removed; 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 node count. However, the node count as defined above *changes* at an eigenvalue. The programs define the node count for an eigenvalue as the node count just above it in energy.

The procedure that BOUND and FIELD use to assign a node count to an eigenvalue 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 eigenvalue itself 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 $100 \times \text{DTOL}$, 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 [15] to converge on a bound state. Convergence terminates when the predicted step size is less than DTOL (in $\&\text{INPUT}$ 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.

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 field. If the absolute value of DFIELD is less than the absolute value of $\text{FLDMAX} - \text{FLDMIN}$, FIELD scans over EFV values in the range FLDMIN to FLDMAX using a step size of DFIELD and report the node count at each step, without attempting to converge on eigenstates. This enables the user to identify field 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 .

10.4 Bound-state wavefunctions

If $\text{IWAVE} \neq 0$ and the LDMD propagator ($\text{IPROPS} = \text{IPROPL} = 6$) is used, then once a bound state has been located, BOUND or FIELD repeats the propagation at the converged energy or field, saving the log-derivative matrices to a temporary file and then propagates the wavefunction calculated at R_{match} back out to the RMIN and RMAX , as described in ref. [16]. Simpson's rule is used to find the normalisation constant.

10.5 Calculating expectation values

In addition to calculating bound-state energies, BOUND implements the calculation of expectation values using the finite-difference approach [56]. 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,

$$E_n(a) = E_n^{(0)} + a\langle\hat{A}\rangle_n + \mathcal{O}(a^2), \quad (10.1)$$

where $\mathcal{O}(a^2)$ are second-order terms. The finite-difference approximation to the expectation value $\langle\hat{A}\rangle_n$ is

$$\langle\hat{A}\rangle_n = \frac{E_n(a) - E_n^{(0)}}{a}, \quad (10.2)$$

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 `IP`th value calculated is the expectation value of

$$\text{FACTOR}(\text{IP}) \times \text{A}(\text{IPPERT}(\text{IP}))/R^{\text{NPOW}(\text{IP})}$$

The program applies a perturbation of `DELTA(IP) × 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.

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 the LDJ, LDMD and LDMA propagators, the error in bound-state energies due to a finite step size is proportional to the fourth power of the step size (in the limit of small steps). 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.

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 data 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 diagonal basis is used at R_{\min} (section 8.4).

`ADIAMX` [**T**]: Controls whether adiabatic basis or diagonal basis is used at R_{\max} (section 8.4).

ALPHA1 [**1.0**]: Start value of step size for VIVS propagator (section 8.3.3).

ALPHA2 [**1.5**]: End value of step size for VIVS propagator (section 8.3.3).

- 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_{\min} (section 8.4).
- BCYCMX** [−1.0]: Controls value used in construction of log-derivative matrix for channels that are closed at R_{\max} (section 8.4).
- BCYOMN** [10⁸]: Controls value used in construction of log-derivative matrix at channels that are open at R_{\min} (section 8.4).
- BCYOMX** [0.0]: Controls value used in construction of log-derivative matrix at channels that are open at R_{\max} (section 8.4).
- DEGTOL** [10^{−10}]: Criterion used to test degeneracy, given in &INPUT energy units; see sections 9.2 and 9.6.
- DELTA** [0.0]: Array of step sizes used in the calculation of expectation values by finite differences (section 10.5).
- DFIELD** [1.0/10³⁰]: Step size for varying EFV (sections 7.1, 7.2, 9.10.2 and 10.3).
- DNRG** [0.0]: Step size for energies (sections 6.2, 9.9).
- DR** [0.02]: Step size for propagation of coupled equations (sections 8.1.3, 8.3.3–8.3.6)
- DRAIRY**: \equiv DRLR
- DRCON** [0.1]: Size of change to RMID or RMAX used to test convergence in MOLSCAT (section 9.5) and BOUND (section 10.6).
- DRLR** [−1.0]: Starting value of DR used for the long-range part of the propagation (if appropriate) (sections 8.1.3, 8.1.4 and 8.3.6).
- DRMAX** [5.0]: Maximum allowed step size for VIVS propagator (section 8.3.3).
- DRNOW**: \equiv 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 field-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_{tot} (section 9.6.3).
- ECTRCT** [−10³⁰]: Energy cutoff for contraction of basis set in R-matrix propagator (section 4.9).
- EMAX** [10³⁰]: Upper end of energy range for calculations in BOUND (sections 6.5, 10.2).
- EMIN** [−10³⁰]: 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).
- EREF** [0.0]: Reference energy used for input and output of energies (scattering energy, depth of bound states) (section 6.6).
- EUNITS** [1]: Units of energy (section 6.1).
- FACTOR** [0.0]: 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(MAXEFV)** [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).
- IABSDR** [0]: Controls whether step size is taken from DR or STEPS in fixed-step-size propagators (section 8.1.3).
- IALFP** [F]: Controls whether ALPHA2 is used as end value for step size in VIVS propagator (section 8.3.3).
- IALPHA** [6]: Controls whether step size can be variable in VIVS propagator (section 8.3.3).
- 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 4.8).
- IBHI** [0]: Highest value of symmetry block index for which calculations are required (section 4.8).
- ICHAN** [1]: Index of open channel to use in characterising a field-dependent Feshbach resonance (section 9.10.3), converging on a field 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).
- IDIAG** [F]: Master control for perturbation correction type in VIVS propagator (section 8.3.3).
- IFCONV** [0]: Specifies a scan of scattering length as a function of field (section 9.10.2), the type of Feshbach resonance for characterisation (section 9.10.3), or convergence on a field with a specified value of the scattering length (section 9.10.4).

IFVARY(MAXEFV) [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 8.3.5).

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 8.3.3).

IPHSUM [0]: Unit number for summary of eigenphase sums and scattering lengths (section 13.2).

IPPERT [0]: Array of indices of operators for which expectation values are to be calculated (section 10.5).

IPRINT [0]: Controls extent of printed output; see chapter 12.

IPROPL [1/6]: 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 8.1.2).

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 8.1.2).

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 8.6).

ISHIFT [F]: Controls whether 2nd derivative is used to shift reference potential for VIVS propagator (section 8.3.3).

- ISIGPR** [0]: Controls printing of cross sections (sections 12.1 and 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 8.3.3).
- IV** [T]: Controls calculation of perturbation corrections due to potential for VIVS propagator (section 8.3.3).
- IVP** [F]: Controls calculation of perturbation corrections due to 1st derivative of interaction potential for VIVS propagator (section 8.3.3).
- IVPP** [F]: Controls calculation of perturbation corrections due to 2nd derivative of interaction potential for VIVS propagator (section 8.3.3).
- 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 4.8).
- JTOTL** [0]: Lowest value of JTOT (section 4.8).
- JTOTU** [0]: Highest value of JTOT (section 4.8).
- 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 processes another complete input data set after the current one (starting with another &INPUT block), or terminate; see page 94.
- 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 4.10).
- 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).
- MAXSTP** [10000]: Largest number of steps allowed for R-matrix propagator (section 8.3.2).
- MHI**: Deprecated synonym for IBHI.

- MMAX** [0]: Highest M value for which IOS cross sections are accumulated (section 4.10).
- 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 fields) 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 4.4.7).
- 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 8.3.7).
- 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).
- NPRT** [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).
- 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 8.3.3) and for calculating nonadiabatic matrix elements with the LDMA propagator (section 8.3.4).
- 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 4.4.7).
- PHIST [0.0]:** Step size for azimuthal angle in surface scattering calculations (ITYPE = 8, section 4.4.7).
- POWRX [3.0/6.0]:** Inverse power of interaction potential used to estimate next step size for AIRY propagator (section 8.3.6).
- PRNTLV:** deprecated synonym for IPRINT.
- RCTRCT [0.0]:** If greater than 0.0, value of R at which basis set is contracted in R-matrix propagator (section 4.9).
- RMATCH [0.0]:** Value of R at which log-derivative matrices for outwards and inwards propagation parts are matched; see sections 8.1.2 and 8.1.4. If left unset, set internally to RMID (if set) or 1.0 (otherwise).
- RMAX [10.0]:** Maximum value of R for propagation (sections 8.1.2, 8.1.4, 8.3.2, 8.3.3 and 8.3.6).
- RMID [1.0D30/0.0]:** Value of R at which propagation method switches for propagations which comprise more than one part (sections 8.1.2, 8.1.4, 8.3.3 and 8.3.6). Also used as the value where the step size starts to increase in the R-matrix propagator (section 8.3.2). If left unset in BOUND or FIELD, set internally to RMATCH (if set) or 1.0 (otherwise).
- RMIN [0.8]:** Minimum value of R for propagation (sections 8.1.2, 8.1.4).
- RVFAC [0.0]:** If set greater than 0.0, controls how RMID 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).
- SHRINK [1]:** Controls removal of channels from basis set for R-matrix propagator (section 8.3.2).
- STABIL [5.0]:** Controls how often linear independence is re-established for de Vogelaere propagator (section 8.3.1).
- STEPS [−10.0]:** If positive, number of steps per half wavelength for wavefunction-following propagators (section 8.1.3). Otherwise DR is used to set the step length for propagations.

- STEST** [0.0001]: Tolerance for terminating propagation for de Vogelaere propagator (section 8.3.1).
- 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 4.4.7).
- THETST** [0.0]: Step size for polar angle in surface scattering calculations (ITYPE = 8, section 4.4.7).
- TOL**: Deprecated synonym for TOLHI.
- TOLHI** [0.0001]: Controls step size for variable-step (VIVS and AIRY) propagators (sections 8.3.3, 8.3.6). Also used as the convergence criterion for the WKB phase shift (section 8.3.7).
- TOLMAX** [1.0]: controls positioning of the outermost of the 3 target points used in characterising a field-dependent Feshbach resonance (section 9.10.3).
- TOLMIN** [0.1]: controls positioning of the second nearest of the 3 target points used in characterising a field-dependent Feshbach resonance (section 9.10.3).
- URED** [no default]: Reduced mass for calculation (chapter 4).
- VTOL** [10^{-6}]: Convergence criterion for R-matrix propagator (section 8.3.2).
- WKBMN** [.TRUE.]: Flag to use WKB boundary conditions in closed channels at R_{\min} (section 8.4).
- WKBMX** [.TRUE.]: Flag to use WKB boundary conditions in closed channels at R_{\max} (section 8.4).
- XSQMAX** [10000]: Controls application of perturbation correction for VIVS propagator (section 8.3.3).

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 4.4.6, 4.4.4 and 4.4.5. 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 vibrator; see sections 4.4.1, 4.4.2, 4.4.3 and 4.4.6.

- B [0.0]:** Array of dimension 2. The rotational constant about the y axis for a symmetric or asymmetric top; see sections 4.4.6, 4.4.4 and 4.4.5. 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 4.11).
- BE [0.0]:** Array of dimension 2. The rotational constant for a linear rotor or vibrotor. See sections 4.4.1, 4.4.2, 4.4.3 and 4.4.6.
- C [0.0]:** Array of dimension 2. The rotational constant about the z axis for a symmetric or asymmetric top; see sections 4.4.6, 4.4.4 and 4.4.5. 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 4.4.1, 4.4.2, 4.4.3 and 4.4.6.
- DJ [0.0]:** The centrifugal distortion constant D_J for a symmetric or asymmetric top; see sections 4.4.4 and 4.4.5.
- DJK [0.0]:** The centrifugal distortion constant D_{JK} ; see sections 4.4.4 and 4.4.5.
- DK [0.0]:** The centrifugal distortion constant D_K ; see sections 4.4.4 and 4.4.5.
- DT [0.0]:** The spherical top tetrahedral centrifugal distortion constant d_t ; see section 4.4.5.
- 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 4.4
- 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 4.4.6, 4.4.5 and 4.4.7.
- EMAXK [0.0]:** If EMAXK > 0.0, it is used to limit the selection of basis functions basis functions for ITYP = 8; see section 4.4.7.
- EUNITS [1]:** Integer denoting energy units used for values input in namelist &BASIS; see section 4.
- IASYMU [0]:** Unit number for input and output of asymmetric top functions; see sections 4.4.6 and 4.4.5.
- 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 4.7. 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 4.4.3.
- IOSNGP [0]:** Array of dimension 3: numbers of quadrature points over angles in IOS calculations; see section 4.10.

- IPHIFL [0]:** Controls the type of quadrature over $\phi_1 - \phi_2$ or χ in IOS calculations; see section 4.10.
- 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 4.4.5, 4.4.6 and 4.4.4.
- 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 4.6).
- 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 4.4.3. 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 4.4.3. 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 4.4.3. 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 4.4.3 and 4.4.6. 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 4.4.3 and 4.4.6.
- J2STEP [1]:** Step used in creating lists of pair levels for second structured particle. See sections 4.4.3 and 4.4.6.
- JHALF [1]:** See section 17.3. Use in namelist deprecated. Should now 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 4.4.
- JMAX [0]:** Maximum value of J used in creating lists of pair levels for first structured particle. See sections 4.4.1, 4.4.2, 4.4.4, 4.4.5 and 4.4.7. 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 4.4.1, 4.4.2, 4.4.4, 4.4.5 and 4.4.7. 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 4.4.1, 4.4.2, 4.4.4, 4.4.5 and 4.4.7. It is equivalenced to **J1STEP** and so may be specified using this name also.

JZCSFL [0]: See section 4.7

JZCSMX [-1]: See section 4.7

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 4.4.4. 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 4.4.4. 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 4.4.3.

WE [0.0]: Array of dimension 2. The equilibrium vibrational frequency of vibrating linear rotors; see sections 4.4.2 and 4.4.6.

WEXE [0.0]: Array of dimension 2. The anharmonicity constant of vibrating linear rotors; see sections 4.4.2 and 4.4.6.

WT [0.0]: Array of dimension 2 used to supply nuclear spin statistics weighting for identical partners; see section 4.4.4.

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 below [`IHOMO2`] and sections 5.1.4 and 5.3.2. 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 molecules 2. For historical reasons, `ICNSYM` may
- IHOMO2 [1]:** also 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
- MMA [-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 [F]:** 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 if `MXLAM` \leq 0, `LVRTP` is forced to be `.TRUE.`.
- 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 $A \exp(-E \times R)$, and if positive it has the form A/R^{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 throughout the programs, including for **RMIN**, **RMAX**, etc. given in **&INPUT**. These units are described as “RM units” in the output. **RM** is specified in units of Å. When a routine **VINIT/VSTAR** or **VRTP** is supplied, **RM** is usually coded there instead of being supplied in namelist **&POTL**. Subsequent calls to **POTENL** handle distances in these units.

Chapter 12

Controlling the print level

12.1 Main output file

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.

In addition, if `IPRINT` ≥ 1, all the programs will also 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 non-diagonal basis sets, 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).

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_{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_{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 \mathbf{V}^A 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 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 Calculation of threshold energies for non-diagonal basis sets

For 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_{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 Propagation(s)

If R_{min} and/or R_{max} have been altered from the input values as described in section 8.1.2, a message is printed to this effect if `IPRINT` ≥ 3 (13 for IOS calculations). More detailed information about the search for a suitable value of R_{min} is printed if `IPRINT` ≥ 8 (18 for IOS calculations) and `IRMSET` > 0 .

R_{min} , R_{max} and the number of steps for each completed propagation segment are printed if `IPRINT` ≥ 3 (13 for IOS calculations) for MOLSCAT and ≥ 8 for BOUND and FIELD.

BOUND and FIELD print the node count for each propagation segment and the number of negative eigenvalues of the matching matrix (section 10.1) if `IPRINT` ≥ 8 . The eigenvalues themselves are printed 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 .

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 8.3.7.
- The AIRY propagator prints the start point, end point, number of intervals and the size of the smallest and largest intervals if `IPRINT` ≥ 13 . In addition, if `IPRINT` ≥ 20 , the midpoints, sizes, and diagonal and off-diagonal correction terms are printed for each interval; see section 8.3.6.
- The de Vogelaere propagator prints the values of R at which the integration starts, at which it has reached the asymptotic limit and at which the S matrix has converged if `IPRINT` ≥ 4 . It will also give a warning if the K matrix is not symmetric if `IPRINT` ≥ 4 . If `IPRINT` ≥ 13 it prints the radial potential coefficients $v_\Lambda(R)$ and the diagonal elements of the (energy-shifted) interaction matrix at the start of the propagation, and also the values of R at which stabilisation is done; see section 8.3.1.
- The R-matrix propagator prints the values of R at which the propagation starts and ends and the number of steps taken if `IPRINT` ≥ 4 . If `IPRINT` ≥ 13 and high-energy channels have been removed it prints the value of R and the new size of the basis. It also prints the largest and smallest eigenvalues of the matching matrix at each step if `IPRINT` ≥ 20 ; see section 8.3.2.
- The VIVS propagator prints the values of R at which the propagation starts and ends and the number of steps taken if `IPRINT` ≥ 4 . It prints the value of R and an estimate of the size of the derivative of the irregular solution $f_2^{i'}(R)$ (as defined in equation 35 of [42]) every time a new interval is started if `IPRINT` ≥ 13 , and all the control data if `IPRINT` ≥ 20 ; see section 8.3.3.
- The quasi-adiabatic log-derivative propagator prints (at each step of the propagation) the lowest 30 adiabats if `IPRINT` ≥ 19 and the first 30 diagonal elements $\hbar^2 \mathbf{W}/2\mu$ if `IPRINT` ≥ 20 , the radial potential coefficients $v_\Lambda(R)/\text{EPSIL}$ if `IPRINT` ≥ 21 , the energy-shifted interaction matrix $\mathbf{W}(R) - \mathcal{E}\mathbf{I}$ (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 .

All the values of `IPRINT` in the above list 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 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.

12.6.3 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**.

If **IPHSUM** > 0, the eigenphase sum is calculated and output on unit **IPHSUM**.

12.6.4 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.

12.6.5 Line-shape cross sections

When calculating line-shape cross sections (section 9.7) without using the IOS approximation, if **IPRINT** ≥ 4 the contributions to the real and imaginary parts of the cross section are printed.

For calculations using the IOS approximation, no additional information is given depending on the value of **IPRINT**.

12.6.6 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):

$$a_L \approx -\frac{\arg S}{2k^n}; \quad \text{and} \quad a_L \approx \frac{(1-S)}{ik^n}. \quad (12.1)$$

12.6.7 Location of energy-dependent resonances in the eigenphase sum

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 the unit **IPHSUM**.

12.6.8 Characterisation of field-dependent resonances

If **IPRINT** ≥ 2 , MOLSCAT prints the position of the characterised resonance, as described in section 9.10.3. If **IPRINT** ≥ 4 , the position, background scattering length/volume and resonance width are printed for every attempt to characterise a resonance. If **IPRINT** ≥ 6 , information is printed about how the field for the next calculation is chosen.

12.6.9 Location of field at which the scattering length/volume has specific value

If **IPRINT** ≥ 4 , information is printed about progress in converging on the field; 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 Location of bound states

If **IPRINT** ≥ 6 , the energy (for **BOUND**) or the field (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** also print the range of each part of the propagation and its contribution to the node count.

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 $5 \leq \text{IPRINT} < 11$ 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 node number and location (energy, external fields) of located bound states to unit **IBDSUM**.

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 sections 9.5 and 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	4.4.6 , 4.4.5
IVLU	MBF	yes	Coupling matrices	13.8
ILDSVU	M	yes	Log-derivative matrix	13.7
ICONVU	M	yes	S matrices for use in convergence runs	9.5
IPSI	MBF	no	Wavefunctions	10.4 , 9.11
ISAVEU	M	yes	S matrices if <code>IPHSUM</code> = 0 K matrices if <code>IPHSUM</code> > 0	13.5 13.5
IPHSUM	M	no	Eigenphase sums and scattering lengths for low-energy scattering channel <code>ICHAN</code>	13.2
IBDSUM	BF	no	Summary of energies and field values for bound states	13.1
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	8.6

The following sections describe these files in more detail, except for those covered elsewhere.

13.1 Summary of bound states (BOUND and FIELD only)

BOUND and FIELD write a concise summary of converged bound-state energies or field on IBDSUM. The summary includes the node 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 RDPCH), 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 2019.0 (beta), 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.

IPROGM is the version number for the format of the output written to channel ISAVEU.

IPROGM is distinct from the program version number and is 17 for MOLSCAT version 2019.0 (beta).

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

NDGVL is the number of diagonal terms in the internal Hamiltonian that depend on EFVs.

NCONST is the number of (non-diagonal) terms making up the internal Hamiltonian.

NRSQ is the number of (non-diagonal) terms used for the operator \mathcal{L} .

IBOUND is non-zero if the matrix elements for the operator \mathcal{L} are just the diagonal values $L(L+1)$.

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 the calculations use a proxy variable, in which case it is equal to 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. (INDLEV(I), L(I), WV(I), I = 1, NOPEN)

INDLEV(I) this is the index of the element of the array ELEVEL that gives the threshold energy for channel I.

L(I) is the orbital angular momentum for channel I. This is either an integer variable L giving the value of L if IBOUND = 0 or a double precision variable CENT giving the diagonal element of the operator \hat{L}^2 otherwise.

WV(I) is the wavevector of channel I (\AA^{-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.)
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

NBLOCK depends on [ITYPE](#). Note that not every S matrix will necessarily exist. 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.

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 [57, 58].

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 with no specific meaning)

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 read 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 data files and their associated outputs, to provide examples of program features and to allow users to verify that their program build is operating correctly.

For each `.input` file described below, a corresponding `.output` file is provided.

We have also provided a basic GNUmakefile which can be used to build the executables required to run the programs below, with some minimal modifications. The GNUmakefile sets up `gfortran` as the compiler in the variable `Compiler`, but this can be changed at the user's discretion. We have tested this GNUmakefile with the `gfortran`, `pgf90` and `ifort` compilers. The `LIBS` variable (which is currently unset) should if possible contain the addresses of LAPACK and BLAS library routines. If this is not possible, the library routines must be downloaded as described in section 15.6, and the object code files included in the variable `LIBUTILS`. Unless otherwise specified, input data files for MOLSCAT, BOUND or FIELD can use any executable for the specified program.

14.1 Examples for MOLSCAT

14.1.1 All available propagators for MOLSCAT

input file: `molscat-all_propagators.input`
executable: any MOLSCAT executable

`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 3.7.1. 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 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: any MOLSCAT executable

`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 data 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 Location of quasibound state (Feshbach resonance) for Ar-HF

input file: `molscat-Ar_HF.input`
executable: `molscat-Ar_HF`

`molscat-Ar_HF.input` demonstrates the procedure for locating a narrow resonance in the S-matrix eigenphase sum as a function of energy, described in section 9.9. It performs calculations on the H6(4,3,2) potential of Hutson [59] for the ground ($v = 0$) vibrational state of HF, using the LDMD propagator. It sets `NNRG` = -10; it first does scattering calculations 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 does another 5 calculations 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. Nevertheless, the 10 points produced from this run are sufficient to determine the resonance parameters accurately when used with programs SAVER and RESFIT.

The extrapolation procedure used 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.

14.1.4 Line-shape cross sections for Ar + CO₂

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 CO₂ in Ar, using the single repulsion potential of Hutson *et al.* [60] 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. [61].

14.1.5 Line-shape cross sections for Ar + H₂

input file: `molscat-Ar_H2.input`
 executable: `molscat-Rg_H2`

`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 BC₃(6,8) interaction potential of Carley and Le Roy [62], evaluated for H₂ states $(j, v) = (0, 0)$, $(2, 0)$ and $(4, 0)$ using H₂ 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 H₂; 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. [26].

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 para-H₂ and ortho-H₂. The interaction potential is that of Zarur and Rabitz [63].

MOLSCAT calculates elastic and state-to-state inelastic cross sections. This calculation requires that contributions from partial waves are accumulated until the partial-wave sums are converged within the limits set by the input data. Two pairs of calculations are performed, first with close-coupling calculations and then with the coupled-states approximation. All calculations use the LDMD/AIRY hybrid propagation scheme.

Each calculation is done twice; one in which the radial potential coefficients are provided, and one in which the unexpanded potential is provided and is expanded by quadrature by the program. The two calculations illustrate the equivalence of the two methods.

14.1.7 ITYP = 5: cross sections for atom + symmetric top collisions, illustrating automated convergence testing

input file: `molscat-ityp5.input`
 executable: any MOLSCAT executable

`molscat-ityp5.input` contains input data for atom + symmetric top collisions between He and ortho-NH₃, taking account of the tunnelling splitting of NH₃. It uses using a simple analytical interaction potential and switches between the LDMD propagator at short range and the AIRY propagator at long range. The data 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 five 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, mid-point and end point of the propagation. The final two calculations carry out full cross-section calculations, using converged values for the propagation variables, using close-coupling calculations and then 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 CH₄, using the interaction potential of Buck *et al.* [64]. The ground-state rotational constants and the tetrahedral centrifugal distortion constant d_t are specified in the data file and the program uses them to calculate properly symmetrised spherical-top wavefunctions. The data file selects CH₄ rotor functions of A symmetry by setting `ISYM` to 224. 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. [65], 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: any MOLSCAT executable

`molscat-ityp8.input` contains input data for diffractive scattering (`ITYPE` = 8) of He from solid LiF, using the model potential of Wolken [27]. 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 ³Σ 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.* [66]. The coupled-channel equations are solved using the LDMD/AIRY hybrid propagation scheme (`IPROPS` = 6, `IPROPL` = 1).

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 data 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 data file requests calculations at kinetic energies of 1, 10 and 100 mK above the $n = 0$, $j = 1$, $m_j = 1$ threshold of NH. This is done using the `LOGNRG` = `.TRUE.` as described in section 6.2.

These calculations are similar to (a subset of) those of Wallis *et al.* [67], although the test run

uses a smaller basis set than ref. [67]. Convergence at 100 mK requires inclusion of incoming partial waves up to $L = 3$, which requires values of M_{tot} from -2 to 4 for incoming $m_j = 1$. This is represented in the data file with `JTOTL = -2`, `JTOTL = 4`.

MOLSCAT can accumulate cross sections from calculations for different values of M_{tot} and total parity at a single field. 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 calculation of effective range for $^{85}\text{Rb}+^{85}\text{Rb}$

input file: `molscat-Rb2.input`
 executable: `molscat-Rb2`

`molscat-Rb2.input` contains input data for low-energy $^{85}\text{Rb}+^{85}\text{Rb}$ collisions in a magnetic field, using the plug-in basis-set suite described in section 18.2 and the potential of Strauss *et al.* [68], implemented with the VINIT/VSTAR routine described in section 19.2. This is the same system used for the basic resonance scan described in section 3.9.1. All the calculations use the LDMD/AIRY hybrid propagation scheme (`IPROPS = 6`, `IPROPL = 1`).

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 [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 [12]. The fourth is subject to strong background inelasticity. Following those 4 calculations, a further calculation obtains the effective range across the strong resonance observed in the aa channel near 850 G, described in section 3.9.1, 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 in the form

$$\lambda(R) = E_h \alpha^2 \left[\frac{g_S^2}{4(R/a_0)^3} + A \exp(-\beta R/a_0) \right] \quad (14.1)$$

and is coded in namelist `&POTL`. The radial coefficient is scaled by $-E_h \alpha^2$ internally, so that the items given in `&POTL` are $A(1) = -g_S^2(a_0/\text{RM})^3/4$, $A(2) = -A$ and $E(1) = -\beta \text{RM}/a_0$.

14.2 Examples for BOUND

14.2.1 All available propagators for bound-state calculations

input file: `bound-all_propagators.input`
 executable: any BOUND executable

`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 $R_{\text{mid}} < R_{\text{match}}$ and once with $R_{\text{mid}} > R_{\text{match}}$. The calculation which uses just the LDMD propagator employs a different step length for the inwards propagation. This data file should produce the same results regardless of which BOUND executable is used.

14.2.2 Bound states of Ar-HCl with expectation values

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 [69] and the LDMD propagator, for total angular momentum $J = 0$ 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. [69].

14.2.3 Bound states of Ar-CO₂ 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 vibrationally excited state of Ar-CO₂, using the split repulsion potential of Hutson *et al.* [60] and the LDJ propagator, for total angular momentum $J = 0$. The results may be compared with Table IV of ref. [60].

It first calculates the ground-state energy using a fairly large (unconverged) step size of 0.03 \AA . 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-H₂

input file: `bound-Ar_H2.input`
executable: `bound-Rg_H2`

`bound-Ar_H2.input` performs close-coupling calculations on the ground state of Ar-H₂ with H₂ in its $v = 1$, $j = 1$ state, for total angular momentum $J = 1$ 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 H₂ ($v = 0$) [70], 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 [71], evaluated for H₂ states $(j, v) = (0, 0)$, $(2, 0)$ and $(4, 0)$ using H₂ 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 H₂-H₂ (ortho-para)

input file: `bound-ityp3.input`
executable: `bound-H2_H2`

`bound-ityp3.input` performs close-coupling calculations on bound states of H₂-H₂ with one para-H₂ molecule (even j) and one ortho-H₂ molecule (odd j). It uses the LDMD propagator. The interaction potential is that of Zarur and Rabitz [63]. The states are bound by less than 1 cm⁻¹ (below the $j=0 + j=1$ threshold).

14.2.6 Bound states of He-NH₃

input file: `bound-ityp5.input`
executable: any MOLSCAT executable

`bound-ityp5.input` performs close-coupling calculations on bound states of He-NH₃, taking account of the tunnelling splitting of NH₃, using a simple analytical interaction potential and the LDMD propagator. The data 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-CH₄

input file: `bound-Ar_CH4.input`
executable: `bound-Ar_CH4`

`bound-Ar_CH4.input` performs close-coupling calculations on bound states of Ar-CH₄, using `ITYPE = 6`, which can also handle complexes of asymmetric tops. It uses interaction potential of Buck *et al.* [64]. It uses the LDMD propagator. The data file selects CH₄ rotor functions of F symmetry by setting `ISYM` to 177. The results may be compared with Table II of ref. [25].

14.2.8 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` contains input data to locate the bound states of MgNH 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.* [66]. 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 will cross threshold near 387 G.

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` contains input data to locate magnetic fields in the range 0 to 400 G at which bound state 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 14.2.8. The coupled equations are solved using the LDMD/AIRY hybrid propagation scheme.

The run locates the same level as in section 14.2.8 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 $^{85}\text{Rb}_2$

input file: `field-basic_Rb2.input`
 executable: `field-Rb2`

`field-basic_Rb2.input` contains input data to locate magnetic fields where bound states cross the lowest scattering threshold for $^{85}\text{Rb}_2$. 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.* [68], 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 3.9.2.

14.3.3 Bound states of $^{85}\text{Rb}_2$ as a function of magnetic field

input file: `field-Rb2.input`
executable: `field-Rb2`

`field-Rb2.input` contains input data to locate bound states of $^{85}\text{Rb}_2$ as a function of magnetic field, using the same potential and basis-set suite as in section 3.9.2. 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 GNUmakefile (`GNUmakefile`) that can build the executables needed for the example calculations described in sections 3.7, 3.8 and 3.9 and in chapter 14;
- a directory `examples` containing
 - a sub-directory `input` containing input data 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 GNUmakefile 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 program

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`, `bd.driver` or `fld.driver`) 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

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

F02AAF Diagonalise symmetric matrix without eigenvectors

F02ABF Diagonalise symmetric matrix with eigenvectors

The programs also call BLAS routines such as DAXPY, DDOT etc. in many places.

15.7 Matrix multiplication

The programs call DGEMUL. This was originally a routine from the IBM ESSL library. In version 2019.0 (beta), 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.

15.8 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 2019.0 (beta) 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 involved save information from one step to the next, and this advantage is lost if the problem is formulated in terms of linear equation solvers.

15.9 Linear equation solver

The programs call the LAPACK routine DGESV directly. The speed of this routine is not critical for most propagators.

15.10 Eigenvalues and eigenvectors of symmetric matrices

These routines are important for propagators 1, 3, 7, and 14 (Airy, R-matrix, quasi-adiabatic log-derivative and VIVS). The programs call diagonalisers by the names of now-withdrawn NAG routines F02AAF and F02ABF. This distribution provide routines that simulate F02AAF and F02ABF by calls to the LAPACK routine DSYEVX.

The versions of F02AAF, F02ABF and SYMINL supplied use allocatable arrays to supply workspace for the LAPACK routines that they call. We have encountered a problem with this in F02ABF with some library versions of DSYEVX: if the workspace is passed by the address of its first element, the results are very slightly different from those obtained if a fixed-size array is used. If, however, it is allocated to be one element longer than it needs to be and is passed by the address of its second element, this discrepancy disappears. We do not know the source of this problem, but we have implemented the workaround described above.

15.11 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 your system. For example, most Linux systems use filename `fort.NN` if unit NN is opened.

15.12 Values of fundamental constants

All three programs use a module `physical_constants`, which contains values of fundamental physical constants and derived constants. The current values are taken from the 2014 revision of the NIST CODATA tables. A message is printed in the output stating the date of the revision. All the values set in this module 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.

A user who is writing their own code may use the module by inserting the line

```
USE physical_constants
```

immediately after the routine declaration statement.

15.13 Array dimensioning

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 program, which calls the relevant driver routine to run one of the programs. The relevant `DRIVER` and other routines then partition the `X` array 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 `X` array 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 `X` array and recompile. 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 `X` array 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 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 `X` array, `CHKSTR` stops the program and prints the message above. Space in the `X` array 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.

15.14 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 nodes 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;
- **MAXSCH** 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 5.4.

The dimension of arrays that are used for information about EFVs depend on a parameter **MAXEFV**, which is set in module **efvs**.

15.15 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**.

CNTR0L 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).

DRIVE contains variables relevant to the propagation.

EIGSUM contains eigenphase sums used for estimating the position of a nearby (energy) resonance.

EXPVAL contains variables relevant to the calculation of expectation values.

HIBRIN contains variables relevant to the Airy propagator.

IOCHAN contains unit numbers for IPSI, IPSISC and IWAISC, 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.

LDVVCN contains variables for the VIVS propagator.

MEMORY is described in section 15.13.

NPOT contains variable for surface scattering calculations.

POPT contains variables to control level of printing for the VIVS propagator.

PRBASE contains variables relevant to calculations of line-shape cross sections.

PRPDTA contains variables relevant to propagation segments.

VLFLAG contains variable indicating whether IV array is used to index VL array.

VLSAVE contains unit number for storage of the VL array.

WKBCOM contains variables relevant to the Gauss-Mehler quadrature performed for WKB integration.

15.16 Testing the installation

After installation, it is highly desirable to validate the programs by running all the supplied test cases and verifying that the programs give essentially identical output.

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. In addition, some quantities are output at close to machine precision, so may have slightly different values with different compilers or on different architectures. It is therefore necessary to exercise some judgement in deciding whether results differ *significantly* from the test output.

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,

$$V(R, \xi_{\text{intl}}) = \sum_{\Lambda=1}^{\text{MXLAM}} v_{\Lambda}(R) V^{\Lambda}(\xi_{\text{intl}}). \quad (16.1)$$

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 (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 is 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 i th column of the **LAMBDA** array should correspond to the i th 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 53. 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 $\text{EPSIL} = 1.0\text{D}0$ 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: IPRINT is 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 $v_{\Lambda}(\text{RR})$ and return them in P

IC = 1 evaluate dv_{Λ}/dR at $R = \text{RR}$ and return them in P

IC = 2 evaluate d^2v_{Λ}/dR^2 at $R = \text{RR}$ 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

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 $\text{EPSIL} \times \text{RR}^{-\text{IC}}$; 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 non-adiabatic couplings).

Even in these cases, there is an option (controlled by logical variable NUMBER 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-skel.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_{intl} and/or \hat{L}^2 is non-diagonal, matrix elements of the operators used to expand them are also required.	17.9

These routines may be (and usually are) supplied as entry points to **BAS9IN**, so that variables

in `SAVE` statements are common to all of them.

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 and 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 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_{intl} and \hat{L}^2 are diagonal;
2. Basis sets in which H_{intl} and/or \hat{L}^2 are non-diagonal.

These will 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 external field(s) usually require non-diagonal basis sets, since the same basis set seldom diagonalises H_{intl} at different values of the field(s).

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_{tot} and IBLOCK for any additional symmetries in the Hamiltonian (total parity, body-fixed K , etc.). However, for calculations in a magnetic field, J_{tot} is not conserved. For a single field (or parallel fields), however, its projection M_{tot} onto the field axis may be conserved; in this case the variable JTOT is conveniently used for M_{tot} .

The programs hold quantum numbers in integer variables and arrays. For systems with half-integer spins, it is 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 $2J_{\text{tot}} + 1$:

JHALF = 1 indicates that JTOT is an undoubled total angular momentum J_{tot} ;

JHALF = 2 indicates that JTOT is a doubled total angular momentum, $2J_{\text{tot}}$;

$\text{JHALF} = 0$ indicates that JTOT is not a total angular momentum, and omits the factor $(2J_{\text{tot}} + 1)$ 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

$$W_{ij}(R) = \frac{2\mu}{\hbar^2} \left(\sum_{\Lambda} v_{\Lambda}(R) \mathcal{V}_{ij}^{\Lambda} + \sum_{\Omega} h_{\Omega} \mathcal{H}_{ij}^{\Omega} \right) + \sum_{\Upsilon} \mathcal{L}_{ij}^{\Upsilon} / R^2. \quad (17.1)$$

All the coupling matrices \mathcal{V}^{Λ} , \mathcal{H}^{Ω} and \mathcal{L}^{Υ} are calculated prior to the propagation by CPL9.

For both diagonal and non-diagonal basis sets, the potential coupling matrices \mathcal{V}^{Λ} 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 \mathcal{H}^{Ω} and/or NRSQ centrifugal matrices \mathcal{L}^{Υ} are also stored in the array VL.

For diagonal basis sets, the coupling matrices \mathcal{H}^{Ω} 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

$$E_{\text{intl},i} = E_{\text{intl},i}^{\text{field-free}} + \sum_{\Omega} h_{\Omega} \mathcal{H}_{ii}^{\Omega}. \quad (17.2)$$

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 \mathcal{L}^{Υ} , 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 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 \mathcal{L}^{Υ} 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 is commonly done using a namelist block **&BASE9**, but this is not compulsory. The corresponding input data must be included in the data 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, so it is usually desirable to include them in **SAVE** statements and implement the other routines that need them as entry points in **BAS9IN**.

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 that coefficients of the centrifugal energies are provided in the array **CENT** returned from **CPL9**. If **NRSQ** ≠ 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** ≠ 0 otherwise.

BAS9IN must set the following quantities that are included in module **potential**:

NCONST: set to 0 if H_{intl} is diagonal in the basis set. For non-diagonal basis sets, the number of terms in the expansion (2.8) of H_{intl} , including terms used for external field variables (see below).

NDGVL: for diagonal basis sets, the number of diagonal terms in the expansion (2.8) of H_{intl} , including terms used for external field variables (see below).

VCONST: array of R -independent coefficients for the terms in the expansion of H_{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_{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.

The internal quantities

$$\text{NEXBLK} = \sum_{i=1}^{\text{NEXTRA}} \text{NEXTMS}(i),$$

$$\text{NPOTL} = \text{MXLAM} + \text{NCONST} + \text{NRSQ}$$

and

$$\text{NVLBLK} = \text{NPOTL} + \text{NEXBLK}.$$

are also included in module `potential`, and are set after the initialisation call to `POTENL`.

`BOUND` includes a facility to calculate expectation values using a finite-difference method [56], as described in section 10.5. To use this, the operator concerned must be part of H_{intl} . If it is not naturally part of H_{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 external field variables.

EFVNAM: array of character strings describing the EFVs, with maximum lengths specified in module `efvs`.

EFVUNT: array of short character strings giving 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.

The namelist item **DEGTOL** is used as a threshold for degeneracy for both H_{intl} and the extra operators. It is treated as an energy and so is scaled by **EUNITS**. 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, NLAV, 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 **BASIN**.

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.

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)

```

C 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)

$$v_{\Lambda}(R) = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} w_{i_1}^{(1)} w_{i_2}^{(2)} f^{\Lambda} \left(\xi_{i_1}^{(1)}, \xi_{i_2}^{(2)} \right) V \left(R, \xi_{i_1}^{(1)}, \xi_{i_2}^{(2)} \right), \quad (17.3)$$

where $\xi_i^{(d)}$ and $w_i^{(d)}$ are the n_d points and weights required for the quadrature over the functions

$f^{\lambda_d}(\xi^{(d)}); f^\Lambda$ 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 $f^\Lambda(\xi_{i_1}^{(1)}, \xi_{i_2}^{(2)}, \dots)$.

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 $\text{NFUN} = \text{MXLAM} \times \text{NPTS}(1) \times \dots \times \text{NPTS}(\text{NDIM})$, 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, NPOTL, 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, NPOTL, 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 array **VL** and **DGVL**. The first **MXLAM** blocks of **VL** must return coupling matrices \mathcal{V}^Λ for the interaction potential. For non-diagonal basis sets, subsequent blocks must return coupling matrices for the **NCONST** R -independent operators and the **NRSQ** centrifugal operators. These are used as described in section 17.4.

The total number of coupling matrices used for calculating the interaction matrix is **NPOTL**. If extra operators are required to resolve threshold degeneracies, as described in section 9.2, they must be returned as **NEXBLK** extra blocks of the array **VL**. The total number of coupling matrices stored in the **VL** array is then $\text{NVLBLK} = \text{NPOTL} + \text{NEXBLK}$. The variables **NEXBLK** and **NVLBLK** are stored in module **potential**.

The elements in the VL array 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
      VL(IPOTL, IRC) = the (IROW,ICOL)-th element of the IPOTL-th coupling matrix
    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 $\hbar^2 L(L+1)/(2\mu R^2)$. 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 $\hbar^2 \text{CENT}/(2\mu R^2)$.

The array IV, used by some of the built-in coupling cases for more sophisticated indexing of VL, is not supported for ITYPE = 9 in version 2019.0 (beta).

For diagonal basis sets, NCONST = 0. CPL9 can nevertheless 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_{tot} and set JHALF = 0. In this case it is usually sufficient to provide a routine that sets DEGFAC to 1.0 regardless of

the values of JJ1 and JJ2. 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 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 **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**).

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

$$H_{\text{intl}}(\xi_{\text{intl}}) = \sum_{\Omega} h_{\Omega} \mathcal{H}_{\text{intl}}^{\Omega}(\xi_{\text{intl}}). \quad (17.4)$$

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

$$\text{VCONST}(i + \text{MAPEFV} - 1) = \text{EFV}(i) \quad \text{for all } i \in [1, \text{NEFV}].$$

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 $\text{MAPEFV} < 0$ or $\text{IFVARY} < 0$. 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 variable 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_{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), 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), RM, EPSIL
```

```
CHARACTER(10)    :: RMUNIT
```

```
CHARACTER(11)    :: EPUNIT
```

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.

RM, RMUNIT, EPSIL and EPUNIT are included in this module so that the values of RM and EPSIL can be available within a plug-in basis-set suite. The character strings RMUNIT and EPUNIT should contain suitable strings to describe these quantities, and may be set either in POTIN9 or in the initialisation call to VRTP or VINIT; values set in an initialisation call overwrite any supplied in POTIN9.

17.12.3 Module efvs

The specification of module `efvs` is

```
INTEGER, PARAMETER :: MAXEFV=10, LEFVN=20, LEFVU=6
```

```

INTEGER          :: NEFV, ISVEFV, MAPEFV, LISTFV(1:MAXEFV+1), NNZRO,      &
                  IEFVST, NEFVP

```

```

DOUBLE PRECISION :: EFV(0:MAXEFV)

```

```

CHARACTER(LEN=1) :: EFVNAME(0:MAXEFV), SVNAME

```

```

CHARACTER(LEN=1) :: EFVUNIT(0:MAXEFV), SVUNIT

```

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. `EFVNAME(0)` and `EFVNAME(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, ISVEFV)` and `NEFVP` is set internally to be `NEFV` unless there is a proxy variable, in which case it is `NEFV+1`.

`SVNAME` and `SVUNIT` are set internally to be `EFVNAME(ISVEFV)` and `EFVUNIT(ISVEFV)` respectively.

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 1S state interacting with a diatomic molecule in a $^3\Sigma$ state, in the presence of an external magnetic field. The internal Hamiltonian H_{intl} is composed of 4 parts, all for the diatomic molecule [72],

$$H_{\text{intl}} = H_{\text{rot}} + H_{\text{spin-rot}} + H_{\text{spin-spin}} + H_Z, \quad (18.1)$$

where

$$H_{\text{rot}} = B_v \hat{n}^2; \quad (18.2)$$

$$H_{\text{spin-rot}} = \gamma \hat{\mathbf{s}} \cdot \hat{\mathbf{n}}; \quad (18.3)$$

$$H_{\text{spin-spin}} = \frac{2}{3} \lambda \sqrt{\frac{24\pi}{5}} \sum_q (-1)^q Y_2^{-q}(r) [\hat{\mathbf{s}} \otimes \hat{\mathbf{s}}]_q^2, \text{ and} \quad (18.4)$$

$$H_Z = -\hat{\boldsymbol{\mu}} \cdot \mathbf{B} \quad (\text{where } \boldsymbol{\mu} = -g_S \mu_B \hat{\mathbf{s}}). \quad (18.5)$$

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. [72] 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`. 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_{tot} , which is not a good quantum number in the presence of a magnetic field. Instead, $M_{\text{tot}} = m_j + M_L$ is a good quantum number and there is a separate set of coupled equations for each value of M_{tot} . The loop over JTOT is used for M_{tot} , which runs from JTOTL to JTOTU.

The spin, s , is the same for all basis functions and M_L is defined by M_{tot} and m_j , so each basis function (for a given M_{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 $H_{\text{spin-spin}}$, 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 and MAPEFV to 1. If IBSFLG = 2, it sets NCONST = 4, indicating that H_{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 external field is held in the 4th of these blocks. 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 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

$H_{\text{intl}} = H_{\text{rot}} + H_{\text{spin-rot}} + H_{\text{spin-spin}}$. **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 **JSINDEX** and **L** arrays for the current combination of M_{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 $M_L = M_{\text{tot}} - m_j$; only values $L \leq |M_L|$ 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 [72] for the explicit expression.

If **IBSFLG** = 2, the next 4 blocks contain the coupling matrices for H_{rot} , $H_{\text{spin-rot}}$, $H_{\text{spin-spin}}$ and H_Z , respectively; together these make up H_{intl} . They are defined with corresponding prefactors h_Ω (as in Eq. 17.4) B_v , $\gamma\sqrt{s(s+1)(2s+1)}$, $\lambda_3^2\sqrt{30}$ (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 \mathcal{H}^Ω 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_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**.

THRS9 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 $H_{\text{spin-spin}}$. 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.f** handles interactions between two alkali-metal atoms in 2S states in a magnetic field, including hyperfine interactions.

The internal Hamiltonian is composed of two parts,

$$H_{\text{intl}} = H_{\text{hyper}} + H_Z, \quad (18.6)$$

where

$$H_{\text{hyper}} = h \sum_{x=A,B} \zeta_x \hat{\mathbf{i}}_x \cdot \hat{\mathbf{s}}_x; \quad H_Z = \mu_B \sum_{x=A,B} (g_{Sx} \hat{\mathbf{s}}_x + g_{nx} \hat{\mathbf{i}}_x) \cdot \mathbf{B}. \quad (18.7)$$

We used 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_{tot} is not a good quantum number in the presence of a magnetic field. Instead, $M_{\text{tot}} = m_{sA} + m_{iA} + m_{sB} + m_{iB} + M_L$ is a good quantum number and there is a separate set of coupled equations for each value of M_{tot} . Since M_{tot} can be half-integer, the loop over JTOT is used for the *doubled* quantum number $2M_{\text{tot}}$, which runs from JTOTL to JTOTU in steps of JSTEP.

For each basis function, M_L is defined by M_{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_{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 [73].

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_{intl} is described using 2 blocks of the array **VL** as described under **CPL9** below. It also sets **MAPEFV** = 2, indicating that the external 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 they have access to the same list of internal quantities. They also have access to quantities read in namelist &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_{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 $M_L = M_{\text{tot}} - m_j$; only values $L \leq |M_L|$ 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

$$\frac{1 \pm (-1)^L P_{AB}}{[2(1 + \delta_{m_{sA}m_{sB}}\delta_{m_{iA}m_{iB}})]^{1/2}} |m_{sA}m_{iA}m_{sB}m_{iB}LM_L\rangle \quad (18.8)$$

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 [74]. The first **NSPIN** = $1 + \text{MIN}(2s_A, 2s_B)$ 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 term, which is R -dependent so is handled as a potential term. The next **NCONST** = 2 blocks contain the coupling matrices for the atomic hyperfine term and Zeeman term, respectively; together these make up H_{intl} . They are defined with corresponding prefactors h_Ω (as in Eq. 17.4) $1./29.99792458$ (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 \mathcal{H}^Ω whose matrix elements are in the array **VL**.

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 scaling of the spin-spin term is described in section 14.1. 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 **IEXTRA**th 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

19.1 Potential energy surfaces for Ar-HF and Ar-HCl

The routines in file `extpot-v.f` evaluate the $H_6(4,3,2)$ potential of Hutson for Ar-HF [59] and $H_6(4,3,0)$ potential of Ar-HCl [69] at fixed values of R and θ . They are called via a version of VRTP (in file `vrtp-j.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 Ar-HF are supplied in the test input file `bound-Ar_HF.input` described above, and a data file for Ar-HCl is available on request.

The 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. [59], and those required for HCl are given in table II of ref. [69].

19.2 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.* [68], 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 data 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	subroutine	found in
VINIT	vstar-Tiemann.f	VSTAR2	vstar-Tiemann.f
VSTAR	vstar-Tiemann.f	POWER	vstar-Tiemann.f
VSTAR1	vstar-Tiemann.f	DPOWER	vstar-Tiemann.f

The functional form in the well region of the potential, between a short-range limit $r_{S,\text{sr}}$ and a long-range limit $r_{S,\text{lr}}$, is a series expansion

$$\sum_{i=0}^{n_{S,\text{exp}}} a_{S,i} [\xi_S(r)]^i, \text{ where } \xi_S(r) = \frac{r - r_{S,\text{m}}}{r + b_S r_{S,\text{m}}} \quad (19.1)$$

with different sets of parameters for each total spin $S = 0$ or 1 . $r_{S,\text{m}}$ is chosen to be near the equilibrium distance of the state with multiplicity $2S + 1$. The potential is extrapolated to short range ($r < r_{S,\text{sr}}$) with the form $A_S + B_S(r/a_0)^{-n_{S,\text{sr}}}$, where a_0 is the Bohr radius, and to long range ($r > r_{S,\text{lr}}$) with $\sum_{i=6}^{n_{\text{lr}}} -C_i/r^i + (-1)^{S+1}V_{\text{exch}}(r)$. The dispersion coefficients, C_i , are common to both curves and the long-range exchange contribution is $V_{\text{exch}}(r) = A_{\text{ex}}(r/a_0)^\gamma \exp(-\beta r/a_0)$.

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 [75] 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.

MATCHD B_S is often chosen to match the radial derivative of the mid-range polynomial at $r_{S,\text{sr}}$. This is done if **MATCHD** is **.TRUE.**; otherwise, B_S is used unchanged. In either case, A_S is chosen to make the potential curve continuous at $r_{S,\text{sr}}$ (but possibly with a derivative discontinuity if **MATCHD** is **.FALSE.**)

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**.

It should be noted that the expansion term $a_{S,0}$ is always chosen to make the potential continuous at $r_{S,\text{lr}}$. There is no general mechanism to make the potential *derivative* continuous at $r_{S,\text{lr}}$, and the potentials usually have derivative discontinuities at that point.

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 1.6. 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 routines to propagate wavefunctions from log-derivative propagators. 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 low-energy Feshbach resonances (both elastic and inelastic).

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 Journal on Scientific Computing* **14**, 936 (1993).
- [11] R. I. McLachlan and P. Atela. ‘The accuracy of symplectic integrators.’ *Nonlinearity* **5**, 541 (1992).

- [12] 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).
- [13] M. D. Frye and J. M. Hutson. ‘Characterizing Feshbach resonances in ultracold scattering calculations.’ *Phys. Rev. A* **96**, 042705 (2017).
- [14] B. R. Johnson. ‘The renormalized Numerov method applied to calculating bound states of the coupled-channel Schrödinger equation.’ *J. Chem. Phys.* **69**, 4678 (1978).
- [15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran*, (1992), (Cambridge University Press, 1992), vol. 2, chap. 9, pp. 352–355.
- [16] 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).
- [17] A. M. Arthurs and A. Dalgarno. ‘The theory of scattering by a rigid rotator.’ *Proc. Roy. Soc., Ser. A* **256**, 540 (1960).
- [18] 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).
- [19] S. Green. ‘Rotational excitation in H₂-H₂ collisions — close-coupling calculations.’ *J. Chem. Phys.* **62**, 2271 (1975).
- [20] S. Green. ‘Comment of fitting *ab initio* intermolecular potentials for scattering calculations.’ *J. Chem. Phys.* **67**, 715 (1977).
- [21] T. G. Heil, S. Green, and D. J. Kouri. ‘The coupled states approximation for scattering of two diatoms.’ *J. Chem. Phys.* **68**, 2562 (1978).
- [22] 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 H₂O-H₂.’ *J. Chem. Phys.* **102**, 6024 (1995).
- [23] S. Green. ‘Rotational excitation of symmetric top molecules by collisions with atoms: Close coupling, coupled states, and effective potential calculations for NH₃-He.’ *J. Chem. Phys.* **64**, 3463 (1976).
- [24] S. Green. ‘Rotational excitation of symmetric top molecules by collisions with atoms. 2. Infinite-order sudden approximation.’ *J. Chem. Phys.* **70**, 816 (1979).
- [25] J. M. Hutson and A. E. Thornley. ‘Atom-spherical top Van der Waals complexes: A theoretical study.’ *J. Chem. Phys.* **100**, 2505 (1994).
- [26] J. M. Hutson and F. R. McCourt. ‘Close-coupling calculations of transport and relaxation cross sections for H₂ in Ar.’ *J. Chem. Phys.* **80**, 1135 (1984).
- [27] G. Wolken Jr. ‘Theoretical studies of atom-solid elastic scattering: He + LiF.’ *J. Chem. Phys.* **58**, 3047 (1973).

- [28] 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).
- [29] H. Rabitz. ‘Effective potentials in molecular collisions.’ *J. Chem. Phys.* **57**, 1718 (1972).
- [30] 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).
- [31] S. Green. ‘Accuracy of decoupled- L -dominant approximation for atom-molecule scattering.’ *J. Chem. Phys.* **65**, 68 (1976).
- [32] A. E. DePristo and M. H. Alexander. ‘Decoupled L -dominant approximation for ion-molecule and atom-molecule collisions.’ *J. Chem. Phys.* **64**, 3009 (1976).
- [33] R. Goldflam, D. J. Kouri, and S. Green. ‘On the factorization and fitting of molecular scattering information.’ *J. Chem. Phys.* **67**, 5661 (1977).
- [34] 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).
- [35] M. H. Alexander. ‘Hybrid quantum scattering algorithms for long-range potentials.’ *J. Chem. Phys.* **81**, 4510 (1984).
- [36] 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).
- [37] 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).
- [38] 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).
- [39] B. R. Johnson. ‘Multichannel log-derivative method for scattering calculations.’ *J. Comput. Phys.* **13**, 445 (1973).
- [40] D. E. Manolopoulos. ‘An improved log-derivative method for inelastic scattering.’ *J. Chem. Phys.* **85**, 6425 (1986).
- [41] D. E. Manolopoulos, M. J. Jamieson, and A. D. Pradhan. ‘Johnson’s log derivative algorithm rederived.’ *J. Comput. Phys.* **105**, 169 (1993).
- [42] 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).
- [43] R. T Pack. ‘Space-fixed vs body-fixed axes in atom-diatomic molecule scattering. Sudden approximations.’ *J. Chem. Phys.* **60**, 633 (1974).

- [44] R. Goldflam, S. Green, D. J. Kouri, and L. Monchick. ‘Effect of molecular anisotropy on beam scattering measurements.’ *J. Chem. Phys.* **69**, 598 (1978).
- [45] J. L. Bohn, M. Cavagnero, and C. Ticknor. ‘Quasi-universal dipolar scattering in cold and ultracold gases.’ *New J. Phys.* **11**, 055039 (2009).
- [46] 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).
- [47] R. Shafer and R. G. Gordon. ‘Quantum scattering theory of rotational relaxation and spectral line shapes in H₂-He gas mixtures.’ *J. Chem. Phys.* **58**, 5422 (1973).
- [48] 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).
- [49] 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).
- [50] 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).
- [51] 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).
- [52] R. Blackmore and S. Green. ‘Polarized D₂ StokesRaman Q branch broadened by He: A numerical calculation.’ *J. Chem. Phys.* **88**, 4113 (1988).
- [53] S. Green. ‘Rotational excitation in collisions between two rigid rotors: Alternate angular momentum coupling and pressure broadening of HCl by H₂.’ *Chem. Phys. Lett.* **47**, 119 (1977).
- [54] S. Green. ‘Rotational excitation of symmetric top molecules by collisions with atoms. ii. infinite order sudden approximation.’ *J. Chem. Phys.* **70**, 816 (1979).
- [55] 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).
- [56] J. M. Hutson. ‘Coupled channel bound state calculations: Calculating expectation values without wavefunctions.’ *Chem. Phys. Lett.* **151**, 565 (1988).
- [57] 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).
- [58] 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).
- [59] J. M. Hutson. ‘Vibrational dependence of the anisotropic intermolecular potential of Ar-HF.’ *J. Chem. Phys.* **96**, 6752 (1992).

- [60] J. M. Hutson, A. Ernesti, M. M. Law, C. F. Roche, and R. J. Wheatley. ‘The intermolecular potential energy surface for CO₂Ar: Fitting to high-resolution spectroscopy of Van der Waals complexes and second virial coefficients.’ *J. Chem. Phys.* **105**, 9130 (1996).
- [61] 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 CO₂Ar.’ *J. Chem. Phys.* **107**, 1824 (1997).
- [62] R. J. Le Roy and J. S. Carley. ‘Spectroscopy and potential energy surfaces of Van der Waals molecules.’ *Adv. Chem. Phys.* **42**, 353 (1980).
- [63] G. Zarur and H. Rabitz. ‘Effective potential formulation of molecule-molecule collisions with application to H₂–H₂.’ *J. Chem. Phys.* **60**, 2057 (1974).
- [64] U. Buck, A. Kohlhasse, T. Phillips, and D. Secrest. ‘Differential energy-loss spectra for CH₄ + Ar collisions.’ *Chem. Phys. Lett.* **98**, 199 (1983).
- [65] W. B. Chapman, A. Schiffman, J. M. Hutson, and D. J. Nesbitt. ‘Rotationally inelastic scattering in CH₄ + He, Ne, and Ar: State-to-state cross sections via direct infrared laser absorption in crossed supersonic jets.’ *J. Chem. Phys.* **105**, 3497 (1996).
- [66] 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).
- [67] A. O. G. Wallis and J. M. Hutson. ‘Production of ultracold NH molecules by sympathetic cooling with Mg.’ *Phys. Rev. Lett.* **103**, 183201 (2009).
- [68] 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 ⁸⁷Rb₂.’ *Phys. Rev. A* **82**, 052514 (2010).
- [69] J. M. Hutson. ‘Vibrational dependence of the anisotropic intermolecular potential of Ar-HCl.’ *J. Phys. Chem.* **96**, 4237 (1992).
- [70] J. M. Hutson, C. J. Ashton, and R. J. Le Roy. ‘Vibrational predissociation of H₂-Ar, D₂-Ar, and HD-Ar van der Waals molecules.’ *J. Phys. Chem.* **87**, 2713 (1983).
- [71] R. J. Le Roy and J. M. Hutson. ‘Potential energy surfaces for H₂ with Ar, Kr and Xe.’ *J. Chem. Phys.* **86**, 837 (1987).
- [72] 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 (³Σ[−]).’ *Phys. Rev. A* **75**, 022702 (2007).
- [73] E. Arimondo, M. Inguscio, and P. Violino. ‘Experimental determinations of the hyperfine structure in the alkali atoms.’ *Rev. Mod. Phys.* **49**, 31 (1977).
- [74] 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 $\frac{1}{2}$.

- [75] 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 input variables

ADIAMN in &INPUT, 77, 78, 94
 ADIAMX in &INPUT, 77, 78, 94
 ALPHA1 in &INPUT, 74, 94
 ALPHA2 in &INPUT, 74, 94, 96
 ALPHAЕ in &BASIS, 40, 41, 46, 101, 104
 AZERO in &INPUT, 88, 95
 A in &BASIS, 42–44, 46, 101, 104
 A in &POTL, 58, 106
 BCT in &BASIS, 52, 102
 BCYCMN in &INPUT, 77, 78, 95
 BCYCMX in &INPUT, 77, 78, 95
 BCYOMN in &INPUT, 77, 78, 95
 BCYOMX in &INPUT, 35, 77, 78, 95
 BE in &BASIS, 40, 41, 46, 52, 102, 104
 B in &BASIS, 42–44, 46, 102, 104
 CFLAG in &POTL, 56, 104
 C in &BASIS, 42–44, 46, 102, 104
 DEGTOL in &INPUT, 63, 81, 82, 95, 109, 145, 151
 DELTA in &INPUT, 92, 95
 DE in &BASIS, 40, 41, 46, 102, 104
 DFIELD in &INPUT, 35, 67, 68, 87, 91, 95, 98
 DJK in &BASIS, 43, 102, 104
 DJ in &BASIS, 43, 102, 104
 DK in &BASIS, 43, 102, 104
 DNRG in &INPUT, 63, 64, 86, 95
 DRAIRY in &INPUT, 95
 DRCON in &INPUT, 81, 92, 95
 DRLR in &INPUT, 71, 72, 76, 95
 DRMAX in &INPUT, 74, 95
 DRNOW in &INPUT, 95
 DR in &INPUT, 70–75, 92, 93, 95, 96, 100
 DTOL in &INPUT, 63, 67, 68, 83, 88, 91, 95, 124
 DT in &BASIS, 44, 102, 104
 ECTRCT in &INPUT, 51, 95
 ELEVEL in &BASIS, 37–42, 44, 46, 48, 102, 135, 143, 146, 156, 157
 EMAXK in &BASIS, 48, 102
 EMAX in &BASIS, 44–48, 102
 EMAX in &INPUT, 31, 63, 64, 70, 90, 95
 EMIN in &INPUT, 31, 47, 63, 64, 90, 95
 ENERGY in &INPUT, 31, 35, 47, 63, 64, 70–72, 85, 86, 88, 96, 99
 EPSIL in &POTL, 57, 59, 61, 79, 105, 110
 EREF in &INPUT, 63, 65, 96, 151
 EUNITS in &BASIS, 39, 44, 102, 151, 155
 EUNITS in &INPUT, 33, 39, 63, 81, 96, 145
 E in &POTL, 58, 106
 FACTOR in &INPUT, 92, 96
 FIELD in &INPUT, 67, 68, 96
 FIXFLD in &INPUT, 67, 68, 96
 FLDMAX in &INPUT, 67, 68, 87, 88, 91, 96, 98, 155, 157
 FLDMIN in &INPUT, 67, 68, 87, 88, 91, 96, 98, 155, 157
 IABSDR in &INPUT, 71, 72, 96
 IALFP in &INPUT, 74, 96
 IALPHA in &INPUT, 74, 75, 96
 IASYMU in &BASIS, 44, 102
 IBDSUM in &INPUT, 96, 98, 113
 IBFIX in &INPUT, 33, 50, 51, 81, 86, 96, 99
 IBHI in &INPUT, 50, 51, 81, 96, 98
 IBOUND in &BASIS, 38, 39, 50, 81, 102, 109
 ICHAN in &INPUT, 87–89, 96, 114, 115, 119
 ICNSY2 in &POTL, 61, 105

- ICNSYM in &POTL, 56, 60–62, 105
 ICONVU in &INPUT, 81
 ICON in &INPUT, 81, 92, 93, 96
 IDENT in &BASIS, 41, 102
 IDIAG in &INPUT, 75, 96, 139
 IFCONV in &INPUT, 67, 87, 88, 96
 IFEGEN in &INPUT, 85, 97
 IFIELD in &INPUT, 67
 IFLS in &INPUT, 97
 IFVARY in &INPUT, 66–68, 97, 98, 152
 IHOMO2 in &POTL, 56, 61, 105
 IHOMO in &POTL, 54–56, 60–62, 105
 ILDSVU in &INPUT, 97, 119
 IMGSEL in &INPUT, 76, 97
 INTFLG in &INPUT, 69, 70, 97
 IOSNGP in &BASIS, 51, 102
 IPARTU in &INPUT, 83, 97
 IPERT in &INPUT, 75, 97
 IPHIFL in &BASIS, 51, 103
 IPHSUM in &INPUT, 86, 97, 98, 111, 112, 114, 115
 IPPERT in &INPUT, 92, 97
 IPRINT in &INPUT, 22, 25, 26, 38, 39, 65, 81–83, 97, 100, 107–113
 IPROPL in &INPUT, 69, 70, 89, 91, 97, 124, 125
 IPROPS in &INPUT, 69, 70, 89, 91, 97, 124, 125
 IPSISC in &INPUT, 97
 IPSI in &INPUT, 97
 IREF in &INPUT, 65, 82, 97, 151
 IRMSET in &INPUT, 30, 70, 73, 82, 97
 IRSTRT in &INPUT, 83
 IRXSET in &INPUT, 70, 71, 82, 97
 ISAVEU in &INPUT, 85, 97, 111
 ISCRU in &INPUT, 72, 79, 86, 97
 ISHIFT in &INPUT, 75, 97, 139
 ISIGPR in &INPUT, 22, 98, 107, 111, 112
 ISIGU in &INPUT, 83, 98, 111, 115
 ISYM(3) in &BASIS, 127
 ISYM(4) in &BASIS, 127
 ISYM2 in &BASIS, 49, 103, 135
 ISYM in &BASIS, 42, 44–46, 103, 123, 124, 127, 135
 ISYM in &INPUT, 75, 98
 ITYPE in &BASIS, 17, 23, 36, 40, 44, 46, 47, 51, 52, 62, 84, 85, 99–103, 116, 118, 121, 122, 124, 127, 160
 ITYPE in &INPUT, 48, 138, 140
 IVLU in &BASIS, 103, 120
 IVMAX in &POTL, 55, 105
 IVMIN in &POTL, 55, 105
 IVPP in &INPUT, 75, 98, 139
 IVP in &INPUT, 75, 98, 139
 IV in &INPUT, 75, 98
 IWAVE in &INPUT, 89, 91, 98, 119
 IWAVSC in &INPUT, 98
 J1MAX in &BASIS, 41, 46, 47, 103
 J1MIN in &BASIS, 41, 46, 103
 J1STEP in &BASIS, 41, 46, 103, 104
 J2MAX in &BASIS, 41, 46, 47, 103, 104
 J2MIN in &BASIS, 41, 46, 103
 J2STEP in &BASIS, 41, 46, 103
 JHALF in &BASIS, 103
 JLEVEL in &BASIS, 37–44, 46–48, 82, 102, 103, 135, 146, 156, 157
 JMAX in &BASIS, 40, 42–44, 46, 50, 103, 156, 157
 JMIN in &BASIS, 40, 42–44, 46, 103, 156
 JSTEP in &BASIS, 40, 42–44, 46, 103, 104, 156
 JSTEP in &INPUT, 46, 50, 83, 98, 158
 JTOTLU in &INPUT, 83
 JTOTL in &INPUT, 33, 46, 50, 51, 81, 83, 86, 98, 125, 156, 158
 JTOTU in &INPUT, 33, 46, 50, 51, 81, 83, 86, 98, 124, 156, 158
 JZCSFL in &BASIS, 50, 104
 JZCSMX in &BASIS, 50, 104
 JZCSMX in &INPUT, 112
 KMAX in &BASIS, 42, 103, 104
 KSAVE in &INPUT, 98
 KSET in &BASIS, 42, 43, 103, 104
 L1MAX in &POTL, 54, 56, 105
 L2MAX in &POTL, 54, 56, 105
 LABEL in &INPUT, 98
 LAMBDA in &POTL, 53–57, 59, 62, 105, 139, 148
 LASTIN in &INPUT, 94, 98
 LINE in &INPUT, 85, 98
 LMAX in &INPUT, 51, 98

- LMAX in &POTL, 54–56, 105
 LOGNRG in &INPUT, 64, 98, 124
 LTYPE in &INPUT, 85, 98
 LVRTP in &POTL, 28, 57, 59, 62, 105, 106, 148
 MAGEL in &INPUT, 67, 98
 MAXSTP in &INPUT, 74, 98
 MHI in &INPUT, 98
 MMAX in &INPUT, 51, 99
 MMAX in &POTL, 54, 56, 105
 MONQN in &INPUT, 33, 65, 99, 151, 157, 160
 MSET in &INPUT, 99
 MXCALC in &INPUT, 91, 99
 MXLAM in &BASIS, 23
 MXLAM in &POTL, 54–57, 62, 105, 106
 MXPHI in &INPUT, 46, 99
 MXSIG in &INPUT, 82, 83, 99
 MXSYM in &POTL, 105
 NCAC in &INPUT, 83, 99
 NCONV in &INPUT, 81, 92, 99
 NEFV in &INPUT, 67
 NFIELD in &INPUT, 67, 81, 99
 NFVARY in &INPUT, 67, 68, 99
 NGAUSS in &INPUT, 64, 99
 NGMP in &INPUT, 77, 99
 NLEVEL in &BASIS, 39–42, 44, 46, 47, 103, 104
 NLPRBR in &INPUT, 84, 85, 97, 99
 NNRGPG in &INPUT, 84, 99
 NNRG in &INPUT, 64, 79, 85–88, 99, 122
 NODMAX in &INPUT, 64, 90, 91, 99
 NODMIN in &INPUT, 64, 90, 91, 99
 NPRT in &INPUT, 92, 99
 NPOWER in &POTL, 58, 106
 NPOW in &INPUT, 92, 99
 NPS in &POTL, 60, 105
 NPTS in &POTL, 60, 105
 NPT in &POTL, 60, 105
 NTEMP in &INPUT, 64, 99
 NTERM in &POTL, 58, 106
 NUMDER in &INPUT, 59, 61, 75, 100, 139
 OTOL in &INPUT, 83, 100, 124
 PHILW in &INPUT, 46, 100
 PHIST in &INPUT, 46, 100
 POWRX in &INPUT, 76, 100
 PRNTLV in &INPUT, 100
 PRTY in &BASIS, 45
 RCTRCT in &INPUT, 51, 100
 RMATCH in &INPUT, 31, 70–72, 76, 91, 100
 RMAX in &INPUT, 69, 70, 72–74, 76, 81, 82, 91, 92, 95, 97, 100, 106
 RMID in &INPUT, 69–72, 82, 95, 100
 RMIN in &INPUT, 69, 70, 72, 81, 82, 91, 92, 100, 106
 RM in &POTL, 57, 59, 61, 70, 79, 106, 125
 ROTI in &BASIS, 42, 47, 104, 135, 146, 155, 161
 RVFAC in &INPUT, 71, 82, 100
 RVIVAS in &INPUT, 71, 100
 SCALAM in &INPUT, 67, 68, 100
 SHRINK in &INPUT, 73, 100
 SPNUC in &BASIS, 41, 104
 STABIL in &INPUT, 73, 100
 STEPS in &INPUT, 72, 96, 100
 STEST in &INPUT, 73, 101
 TEMP in &INPUT, 64, 101
 THETLW in &INPUT, 46, 101
 THETST in &INPUT, 46, 101
 TOLHI in &INPUT, 71, 74–76, 101
 TOLMAX in &INPUT, 88, 101
 TOLMIN in &INPUT, 88, 101
 TOL in &INPUT, 77, 101
 URED in &INPUT, 46, 101
 VTOL in &INPUT, 73, 101
 WEXE in &BASIS, 41, 104
 WE in &BASIS, 41, 104
 WKBMN in &INPUT, 77, 78, 101
 WKBMX in &INPUT, 77, 78, 101
 WT in &BASIS, 41, 104
 XSQMAX in &INPUT, 75, 101