ELSEVIER

Computer Physics Communications 114 (1998) 120–128

Computer Physics Communications

The UK molecular R-matrix codes

Lesley A. Morgan [a], Jonathan Tennyson [b], Charles J. Gillan [c,1]

[a] *Computer Centre, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK*

[b] *Department of Physics and Astronomy, University College London, London, WC1E 6BT, UK*

[c] *Department of Applied Mathematics and Theoretical Physics, The Queen's University of Belfast, Belfast, BT7 1NN, UK*

Received 31 March 1998

Abstract

The R-matrix method has been used with great success in recent years to model low energy scattering of electrons and positrons by molecules. The codes, developed by a consortium of UK research groups as part of the Collaborative Computational Project 2 (CCP2), have now reached a high level of robustness and stability. Their overall design is described in this paper. © 1998 Elsevier Science B.V.

PACS: 34.80.-i; 34.80.Bm; 34.80.Gs

Keywords: R-matrix; Electron scattering; Molecules

1. Introduction

The R-matrix method is now established as arguably the most versatile and efficient method for the modelling of low energy scattering by electrons (and positrons) by atomic and molecular targets. The method itself has been described many times in the literature and has been comprehensively reviewed by Burke and Berrington [1]. In recent years a consortium of people based at Royal Holloway, University of London, University College London, Queen's University Belfast and the CLRC Daresbury Laboratory have applied the method to scattering by diatomic and, more recently, polyatomic molecules. The results of this work have been reviewed by Tennyson [2] and by Tennyson and Morgan [3]. The computational methods have been described in some detail in the book "Computational Methods for Electron Molecule Collisions" (Huo and Gianturco, eds.) [4]. The purpose of this paper is to describe the package of computer programs that has been developed by the group. The programs are available via the Web as part of the Collaborative Computational Project 2 (CCP2) virtual program library (url = `http://patiala.phys.strath.ac.uk/iq/prog.html`).

The modelling of low energy electron scattering by a complex target has much in common with bound state calculations for the system of target plus electron. For example, if the target has no residual charge, then the solution of the scattering problem in the region inside the R-matrix sphere is equivalent to finding bound states of the anion confined within this sphere. For this reason we have chosen to base our scattering codes on quantum chemistry codes. We need to calculate the Hamiltonian matrix elements for a finite sphere, whereas quantum chemistry codes evaluate the integrals over all space. Our strategies for tackling this ma-

[1] Present address: Department of Electrical and Electronic Engineering, The Queen's University of Belfast, Belfast, BT7 1NN, UK.

PII S0010-4655(98)00056-3

jor problem are described in Section 3. We also have to choose a basis to describe the continuum electron and will, in general, require higher angular momenta than is usual in bound state calculations. Quantum chemists are usually interested only in the lowest few eigenstates of a system and modern codes often exploit the 'direct' methods which have been developed for this scenario. In contrast, the R-matrix method, as usually formulated, requires all eigenvalues and eigenvectors of the Hamiltonian matrix. We also need to keep close control over the configurations included in our model. Only one electron can occupy a continuum orbital and we need to ensure that any couplings between target electrons which have been omitted from the target wavefunction do not get reintroduced into the final wavefunction. Modern methods, such as the graphical unitary group approach (GUGA), are of no assistance here. For these reasons we have chosen two distinctly traditional packages as the basis of the codes described here.

The electron scattering code, described in this paper, is based primarily on the Alchemy I code of McLean and others [5,6]. It uses Slater type orbitals (STOs) to represent the atomic orbitals and is, in practice, limited to diatomic targets. Since most integrals are done by numerical quadratures, it also allows us great freedom in our choice of continuum orbitals. In order to study polyatomic systems we need a different approach. Gaussian type orbitals (GTOs) are more appropriate than STOs since the multicentre integrals, which need to be evaluated when building the Hamiltonian, can be done in closed form. This restricts our choice of continuum basis, since these too must be represented by GTOs. Our polyatomic code uses parts of the 'Molecule-Sweden' package of Almlöf and Taylor [7,8] to evaluate integrals, but keeps the same configuration generator and Hamiltonian builder as the diatomic code.

The overall structure of the codes is shown in Fig. 1. We have endeavoured to keep the amount of data needed to be input by the user to a minimum and have used the FORTRAN `namelist` wherever practicable. The original quantum codes were highly machine dependent. We have reduced this dependency to a minimum and have isolated all the remainder, which primarily relate to the default integer length, to a few utility routines which are found in the source module MCDEP. Several modules utilize the Nag library [9]. We have, in general, not retained the names of the Alchemy or Molecule-Sweden originals. Where two modules have similar functionality, we have prefixed the Molecule-Sweden variant by 'SW'. Wherever possible, we have avoided the use of fixed dimension arrays and the use of `common` blocks. Instead we use dynamic core allocation, either via the Fortran90 `allocate` statement, or in the case of Fortran77 codes, via a machine dependent routine which allocates a large array which is subsequently subdivided and used as work space. This strategy means that the codes use only the memory that they require and do not have to be recompiled each time a larger problem is tackled.

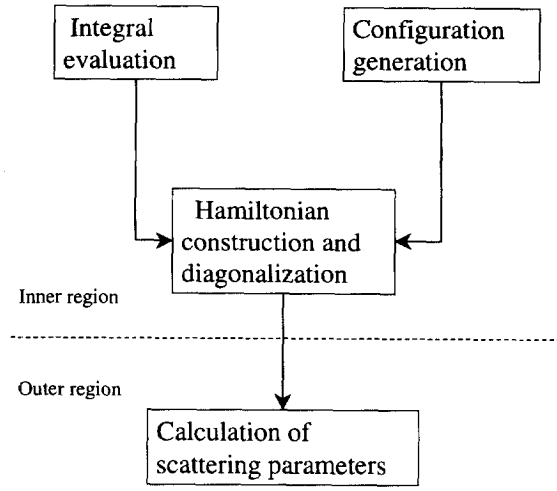

Fig. 1. Code structure for a typical scattering calculation.

2. Wavefunctions

The success or failure of a calculation depends entirely on the choice of trial wavefunction, since the R-matrix method contains no adjustable parameters. The basic form used at each geometry is

$$\psi_k = \sum_{ij} \phi_i(x_1 \ldots x_N) u_{ij}(x_{N+1}) a_{ijk} + \sum_i \chi_i(x_1 \ldots x_{N+1}) b_{ik}, \quad (1)$$

where ϕ_i are target wavefunctions and $u_{ij}(x)$ are continuum orbitals. The χ_i are multi-centre quadratically integrable functions, constructed from the target occupied and virtual molecular orbitals, and are used to

represent correlation and polarization effects. The a_{ijk} and b_{ik} are variational coefficients.

The most important element is the target wavefunctions. Early calculations used only SCF single configuration wavefunctions, but in order to treat electronic excitation multiconfiguration CI wavefunctions are required. Scattering calculations usually do not require highly accurate target energies since only relative energies are important. Rather, it is more intangible things, such as the shape of the target charge cloud, which affect electron scattering. We therefore need good representations of parameters such as permanent dipoles, transition moments and polarizabilities. Calculation of these properties is the function of DENPROP (see below).

The choice of continuum basis depends, as indicated above, on which code is used to evaluate the spatial integrals. If the Alchemy based code is used, then we have complete freedom in our choice of continuum basis, subject only to any constraints on the size of model. A more serious problem is that of linear dependence between this basis and the target orbitals, but experience has shown that this problem is best tackled at the molecular orbital, rather than atomic orbital, level. The Alchemy style continuum orbitals are generated numerically. They are eigensolutions of a simple potential problem with fixed logarithmic boundary conditions at the R-matrix boundary. This boundary condition is introduced in order to obtain an orthonormal set of solutions, complete up to some chosen energy threshold. It imposes an unnecessary constraint on the wavefunction which must be corrected when the R-matrix is eventually constructed. These corrections are known as 'Buttle corrections' [1].

As noted above, the advantage of Gaussian type orbitals is that multicentre integrals can be evaluated in closed form. In order not to lose this advantage and to avoid a major rewriting of the integrals code, we must also represent the continuum orbitals in terms of GTOs. Pioneering work was done by Nestmann and Peyerimoff [13] on this problem and, to date, we have used only the bases which they have constructed. Work on this problem is currently in progress at UCL [14]. These functions are not chosen to satisfy any specific boundary condition at the R-matrix boundary and thus no Buttle correction is needed. The calculation of boundary amplitudes is quite trivial, so no extra data has to be propagated to other modules.

The above discussion relates only to fixed geometry calculations. We have also developed a methodology to describe nuclear vibrations of diatomic systems in a non-adiabatic model also based on an R-matrix treatment [4,15]. This requires data from a sequence of fixed geometry calculations on a grid of internuclear distances, but does not require any changes in the inner region codes. We need to choose a basis to represent the nuclear wavefunction of the vibrating system over a finite range of internuclear distances, but unlike the electronic counterpart, this is not a critical part of the model. Any orthonormal basis which adequately spans the space will do, and in practice we use shifted Legendre polynomials.

3. Integrals

The evaluation of spatial integrals is quite different depending on whether STOs or GTOs are used for the atomic orbitals, though the basic steps remain the same. First, all integrals involving all of the atomic orbitals, target and continuum, are evaluated over the finite sphere. Next continuum molecular orbitals are constructed so as to be orthogonal to the target molecular orbitals and finally four-index transformations are carried out to convert the atomic integrals to molecular integrals.

3.1. STO basis

The modules used by this variant of the code are shown in Fig. 2.

3.1.1. NUMBAS

This module generates numerical continuum orbitals [12]. They are eigensolutions of a simple potential problem with fixed logarithmic boundary conditions at the R-matrix boundary. In practice, it has been found that the model potential can be set to zero for neutral targets and to a simple point charge for ionic targets. In this case the continuum wavefunctions become simply spherical Bessel or Coulomb functions with the given boundary conditions. The amplitudes of the scattering orbitals at the R-matrix boundary are needed to construct the R-matrix and these are stored along with the Buttle corrections for later use.

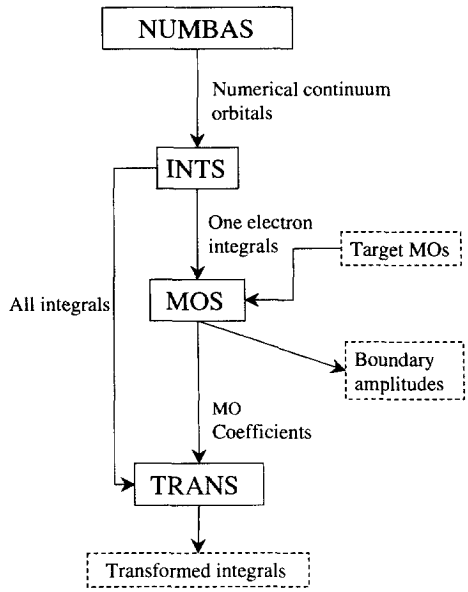

Fig. 2. Modules used to evaluate integrals when STO bases are used.

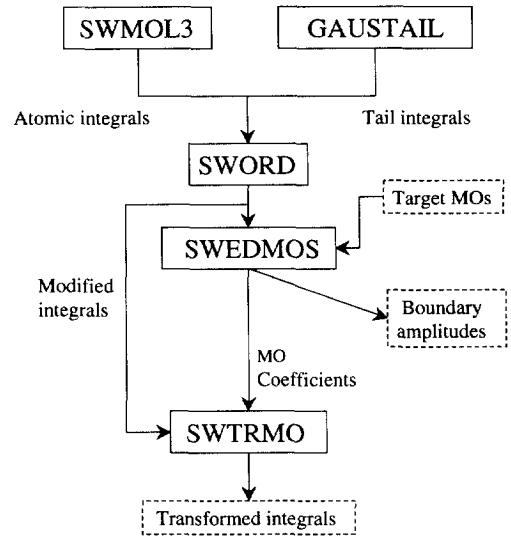

Fig. 3. Modules used to evaluate integrals when GTO bases are used.

3.1.2. *INTS*

This program evaluates all one and two electron atomic integrals and, optionally, property integrals. The modification of the Alchemy original, "LINTP", was carried out by Noble [16]. The integrals are mostly evaluated by numerical quadrature and so it was relatively straightforward to restrict the range of integration to the finite sphere. It was also possible to discard any two electron integrals which have more than one electron in a continuum orbital.

3.1.3. *MOS*

The next step, the construction of continuum molecular orbitals (MOs), has been the subject of much work. The basic problem is to construct continuum molecular orbitals which are linearly independent of the target molecular orbitals without discarding too much of what was originally a basis constructed to span the space of interest. The current code uses a mixture of Schmidt and Lagrange orthogonalization schemes [17]. This module is also used to orthogonalize target virtual orbitals prior to their use in a target CI calculation.

3.1.4. *TRANS*

This program first orders the atomic integrals, then carries out the four-index transformation (i.e. multiplies the atomic integrals by the molecular orbital coefficients) to obtain molecular integrals. It is an adapted version of the Alchemy I module 'TRANSM'.

3.2. *GTO basis*

The modules used in this case are based on the Molecule-Sweden code of Almlöf and Taylor [7,8] and are shown in Fig. 3.

3.2.1. *SWMOL3*

Several variants of the integral generator 'MOL3' were available to us, but all were heavily machine dependent. We started with a Cray version and made it portable. Otherwise the code is unchanged from the original.

3.2.2. *GAUSTAIL*

It was not considered practicable to modify the integrals code to restrict the range of integration. Rather, a separate module was written to evaluate the contribution to each integral from outside the R-matrix sphere. These can also be evaluated in closed form [19]. A consequence of our choice of continuum orbitals is that the Hamiltonian matrix is not necessarily Hermi-

tian. We therefore have to add matrix elements of a Bloch operator [1]. These are merely surface terms which are trivial to evaluate and are included in the 'tail' integrals.

3.2.3. SWORD

The ordering of the integrals is carried out in a separate module, so prior to the actual ordering, we subtract the 'tail' integrals from the atomic integrals output from SWMOL3.

3.2.4. SWEDMOS

The construction of continuum molecular orbitals also differs from the STO case. Here we first use the Schmidt method to orthogonalize each continuum orbital to all of the target orbitals then use the symmetric method to orthogonalize the continuum orbitals amongst themselves, discarding any which correspond to very small eigenvalues of the overlap matrix. Neither this procedure, nor the Lagrange method used in the STO case is restricted to one type of orbital. At present we are evaluating their relative merits.

3.2.5. SWTRMO

The four-index transformation code is unaltered from the original.

4. Configuration generation: CONGEN

This stage, the selection of the configurations which are to be included in the Hamiltonian matrix, is the most crucial to any calculation. It is here that the precise form of the wavefunction of Eq. (1), and hence the model to be used, is specified. The code is a heavily modified version of the Alchemy original [20] and can generate configurations in either the abelian point group D_{2h} and its subgroups, or the non-abelian groups $D_{\infty h}$ and $C_{\infty v}$. A particular modification gives the ability to specify the symmetry of the target wavefunction, independent of the symmetry of the overall system. The code has also been adapted to allow for phase problems which can arise when using CI representations of the target [21].

5. Hamiltonian construction and diagonalization: SCATCI

This module both constructs and diagonalizes the Hamiltonian matrix. For large calculations, this part of the calculation is the slowest.

When CI targets are used, the basic Hamiltonian can become very large, though the required Hamiltonian, obtained by multiplying the original by the CI expansion coefficients, is almost always very much smaller. A special algorithm has been developed [22] which takes advantage of the structure of the scattering wavefunction, Eq. (1). This algorithm has speeded up the Hamiltonian construction phase and has made the use of much larger CI target expansions feasible. It has however shifted the problem to one of diagonalizing very large Hamiltonian matrices [23].

SCATCI is also designed to generate CI target wavefunctions. For this case, when only a few eigenfunctions are required from a large matrix, the Davidson diagonalization procedure of Stathopoulos and Fischer [25] has been implemented as an option.

CONGEN and SCATCI have both been adapted to allow for positron–molecule as well as electron–molecule collisions [24]. At present only the integrals generated by the linear molecule modules are produced in a form appropriate for such calculation.

6. Target wavefunctions and properties

Many of the above modules are used in the calculation of target wavefunctions, however there are others which relate only to the construction of target wavefunctions and the calculation of properties derived from them such as permanent dipoles, transition moments and polarizabilities. A flow chart of a typical target calculation is shown in Fig. 4.

6.1. SCF and SWSCF

We use two different SCF modules, depending on whether our integrals were evaluated using Alchemy or Molecule-Sweden derived codes. They are virtually unaltered versions of the modules in the original packages.

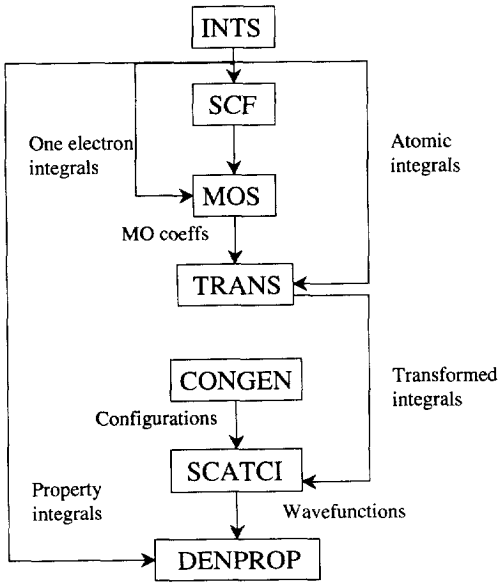

Fig. 4. Modules required for a target state calculation.

6.2. GAUSPROP

This code evaluates the property integrals required by DENPROP if the Sweden-Molecule based codes are being used. It is necessary since, unlike its Alchemy equivalent, MOL3 does not provide them.

6.3. DENPROP

This module calculates properties such as permanent dipoles, transition moments and polarizabilities from input target wavefunctions. It can also used for calculating transition moments between scattering states either for photoionization (bound–free) [10] or bound–bound studies [11].

6.4. PSN

Pseudo natural orbitals are calculated by diagonalizing the density matrices provided by DENPROP.

7. Outer region

The codes, which solve the scattering problem in the region outside the R-matrix sphere and which extract the physical properties from the model, are maintained in the form of a library of relocatable subroutine modules. Different applications will require different subsets of the available modules and the user is required to provide a minimal main program which merely makes calls to the required driving routine for each module. The modules communicate via external files. These files can consist of several stacked datasets, each with their own header information. For example, the T-matrix file can consist of sets of data computed for the same range of energies but different scattering symmetries, or, alternatively, the same symmetry but a range of geometries.

7.1. INTERF and SWINTERF

The most important module is that which interfaces the inner and outer regions. There are two variants depending on whether the inner region Hamiltonian matrix was obtained using the Alchemy or Molecule based integrals codes. The required input is data from the inner region, boundary amplitudes, eigenvalues and eigenvectors of the Hamiltonian matrix and possibly Buttle corrections, and information about the target, energies, multipole moments, etc. The output consists of two files. The first contains parameters which describe the model such as target properties, channel quantum numbers and the overall symmetry of the system. The other contains the basic ingredients for the construction of the R-matrix and the coefficients of a multipole expansion of the long range scattering potentials. This data completely describes the model and is much more compact than the original input data and so it is recommended that these two files are saved for future use. The two variants of the code use the same output format and hence subsequent modules are compatible with either inner region package.

7.2. RSOLVE

The scattering energies are introduced for the first time as input to this module. It constructs the R-matrix, solves the outer region scattering equations and constructs K-matrices for the specified energies. At this stage the fact that the target is a molecule becomes largely irrelevant, since the equations to be solved represent scattering from a single centre. We use a propagator method [27] to propagate the original R-matrix out to a radius where an asymptotic expansion [26]

of the scattered electron wavefunction can be used. A standard formula then gives the K-matrix.

7.3. BOUND

This module has much in common with RSOLVE. However its function is to solve the scattering problem with bound state boundary conditions and thus obtain true bound state energies and wavefunctions of the target + electron system. The algorithm used is a generalisation of one originally developed for atomic problems [28,29].

7.4. VIBRMT

This is an optional module for setting up a non-adiabatic treatment [4,18] of nuclear vibration. The current implementation is for diatomic systems only.

7.5. TMATRX

The T-matrix is calculated from the K-matrix using the standard formula. If required, this module will also perform an adiabatic average to obtain a vibrationally resolved T-matrix.

7.6. IXSECS

This module calculates integrated cross sections from the T-matrix using standard formulae.

7.7. RATES

RATES integrates the calculated excitation cross section over a Maxwellian electron distribution to give temperature dependent rates. Optionally the high energy tail can be allowed for by extrapolation [33].

7.8. EIGENP

Eigenphases are obtained by diagonalizing the K-matrix.

7.9. RESON

RESON [30] searches the eigenphases sums for resonances. Those detected are fitted to a Breit–Wigner profile using an automatically generated grid of points.

7.10. TIMEDEL

TIMEDEL [31] automatically searches for resonances and fits those detected using the time delay method [32]. Branching ratios for autoionization can also be obtained.

7.11. MCQD

The multichannel quantum defect method [34,35] is used to obtain quantum defects for scattering from ionic targets.

7.12. ROTIONS

ROTIONS [36] calculates electron impact rotational excitation cross sections for linear molecular ions using a combination of data from TMATRX and the Coulomb–Born approximation [37].

7.13. TDIP

TDIP takes the bound state wavefunction coefficients generated by BOUND and the transition moments generated by DENPROP and multiplies them together to produce transition dipoles [11].

7.14. DCS

This is the differential cross section code of Malegat [38] interfaced to the OUTER package. It is only appropriate for $C_{\infty v}$ and $D_{\infty h}$ symmetries.

7.15. Examples

A typical outer region run to calculate integrated cross sections for one symmetry and geometry will require a program which makes the following subroutine calls:

```
call INTERF(bigvec,lbigvec,ifail)
call RSOLVE(bigvec,lbigvec,ifail)
call TMATRX(bigvec,lbigvec,ifail)
call IXSECS(bigvec,lbigvec,ifail)
```

The call to each driving routine is similar. The first argument `bigvec` is a large array, declared earlier, which is used for dynamic storage allocation within the module, `lbigvec` is the size of this array in 64 bit words and `ifail` is a success/failure flag (0 indicates success, 1 indicates failure).

If differential cross sections are required, then a loop over scattering symmetries is needed. If the output of INTERF has been saved from previous runs and `nsym` is the number of symmetries, then all that is required is

```
do i=1,nsym
call RSOLVE(bigvec,lbigvec,ifail)
call TMATRX(bigvec,lbigvec,ifail)
end do
call DCS(bigvec,lbigvec,ifail)
```

The input to TMATRX will instruct it to save all T-matrices to a single file which is then read by DCS.

8. Conclusion

The molecular R-matrix codes have reached a level of stability and robustness where they can be used to solve a wide variety of low energy electron and positron scattering problems involving light molecular targets. Our Web pages, accessible via the CCP2 virtual program library (url = `http://patiala.phys.strath.ac.uk/iq/prog.html`), give access to source codes, documentation and example input data.

No provision has yet been made for relativistic effects. The coding should scale to larger problems without difficulty, but inevitably there comes a point where different techniques should be investigated. Linear dependence becomes a problem when large bases are used, especially when diffuse orbitals are included in the target basis. Elaborate models can lead to very large Hamiltonian matrices and it is not practicable in such cases to obtain all eigenvalues and eigenvectors. The crucial factor in any calculation is the choice of trial wavefunction, Eq. (1). We still lack sufficient experience to be able to give reliable guidelines or an algorithmic approach for its construction in any but the most simple systems.

Acknowledgements

We are grateful to the many people who have contributed to the codes outlined above. These include Cliff Noble, Keith Berrington, Susan Branchett, Grahame Danby, Katrina Higgins, Ismanuel Rabadán, Stefano Salvini, Baljit Sarpal and Darian Stibbe. The advice and encouragement of Phil Burke is also gratefully acknowledgement. We thank the UK Engineering and Physical Science Research Council for support under various grants and via Collaborative Computational Project 2 (CCP2).

References

[1] P.G. Burke, K.A. Berrington, R-matrix Theory of Atomic and Molecular Processes (IOP Publishing, Bristol, 1993).
[2] J. Tennyson, in: Proc. XIXth Int. Conf. on Physics of Electronic and Atomic Collisions, L.J. Dubé, J.B.A.Mitchell, J.W. McConkey, C.E. Brion, eds. (AIP Conf. Proc., New York, 1995) p. 233.
[3] J. Tennyson, L.A. Morgan, Proc. Roy. Soc (1998), to be published.
[4] W.M. Huo, F.A. Gianturco, eds., Computational Methods for Electron Molecule Collisions (Plenum, New York, 1995).
[5] A.D. McLean, in: Conference on Potential Energy Surfaces in Chemistry, W.A. Lester Jr., ed. (IBM Research Laboratory, San Jose, 1971) p. 87.
[6] A.D. McLean, M. Yoshimine, B.H. Lengsfield, P.S. Bagus, B. Liu, in: Modern Techniques in Computational Chemistry: MOTECC-91, E. Clementi, ed. (Escom, Leiden, 1991) Ch. 6.
[7] J. Almlöf, P.R. Taylor, in: Advanced Theories and Computational Approaches to the Electronic Structure of Molecules, C.E. Dykstra, ed. (Reidel, Dordrecht, 1984).
[8] J. Almlöf, P.R. Taylor, in: Modern Techniques in Computational Chemistry: MOTECC-91, E. Clementi, ed. (Escom, Leiden, 1991) Ch. 6A.
[9] NAG Fortran Library Manual, Mark 17 (1996).
[10] J. Tennyson, C.J. Noble, P.G. Burke, Int. J. Quantum Chem. 29 (1986) 1033.
[11] S.E. Branchett, J. Tennyson, J. Phys. B 25 (1992) 2017.
[12] S.A. Salvini, PhD Thesis, University of Belfast (1984).
[13] B.M. Nestmann, S.D. Peyerimhoff, J. Phys. B 23 (1990) L773.
[14] A.J. Richardson, J. Tennyson, private communication.
[15] B.I. Schneider, M. Le Dourneuf, P.G. Burke, J. Phys. B 12 (1979) L365.
[16] C.J. Noble, Daresbury Laboratory Technical Memorandum DL/SCI/TMT33T (1982).
[17] J. Tennyson, P.G. Burke, K.A. Berrington, Comput. Phys. Commun. 47 (1987) 207.
[18] C.J. Gillan, O. Nagy, P.G. Burke, L.A. Morgan, C.J. Noble, J. Phys. B 20 (1987) 4585.

[19] L.A. Morgan, C.J. Gillan, J. Tennyson, X. Chen, J. Phys. B 30 (1997) 4087.
[20] R.K. Nesbet, Variational Methods in Electron–Atom Scattering (Plenum, New York, 1980).
[21] J. Tennyson, Comput. Phys. Commun. 100 (1997) 26.
[22] J. Tennyson, J. Phys. B 29 (1996) 1817.
[23] J. Tennyson, J. Phys. B 29 (1996) 6185.
[24] G. Danby, J. Tennyson, J. Phys. B 23 (1990) 1005.
[25] A. Stathopoulos, C.F. Fischer, Comput. Phys. Commun. 79 (1994) 268.
[26] C.J. Noble, R.K. Nesbet, Comput. Phys. Commun. 33 (1984) 399.
[27] L.A. Morgan, Comput. Phys. Commun. 31 (1984) 419.
[28] B.K. Sarpal, S.E. Branchett, J. Tennyson, L.A. Morgan, J. Phys. B 24 (1991) 3685.
[29] I. Rabadán, J. Tennyson, J. Phys. B 29 (1996) 3747.
[30] J. Tennyson, C.J. Noble, Comput. Phys. Commun. 33 (1984) 421.
[31] D.T. Stibbe, J. Tennyson, Computer Phys. Commun. 114 (1998) 236, this issue.
[32] D.T. Stibbe, J. Tennyson, J. Phys. B 29 (1996) 4267.
[33] B.K. Sarpal, J. Tennyson, Mon. Not. R. Astr. Soc. 263 (1993) 909.
[34] M.J. Seaton, Rep. Prog. Phys. 46 (1983) 167.
[35] J. Tennyson, J. Phys. B 21 (1988) 805.
[36] I. Rabadán, J. Tennyson, Comput. Phys. Commun. 114 (1998) 129, this issue.
[37] I. Rabadán, B.K. Sarpal, J. Tennyson, J. Phys. B, in press.
[38] L. Malegat, Comput. Phys. Commun. 60 (1990) 391.