QUEEN'S UNIVERSITY BELFAST

2DRMP: A suite of two-dimensional R-matrix propagation codes

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

40th Anniversary Issue

2DRMP: A suite of two-dimensional R-matrix propagation codes [☆]

N.S. Scott [a,*], M.P. Scott [b], P.G. Burke [b], T. Stitt [c], V. Faro-Maza [a], C. Denis [d], A. Maniopoulou [e]

[a] School of Electronics, Electrical Engineering & Computer Science, The Queen's University of Belfast, Belfast BT7 1NN, UK
[b] School of Mathematics & Physics, The Queen's University of Belfast, Belfast BT7 1NN, UK
[c] CSCS, Swiss National Supercomputing Centre, Galleria 2 - Via Cantonale, CH-6928 Manno, Switzerland
[d] EDF, EDF Research and Development, SINETICS Department, 1 avenue du Général de Gaulle, 92141 Clamart Cedex, France
[e] The Numerical Algorithms Group Ltd, Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, UK

ARTICLE INFO

Article history:
Received 1 April 2009
Received in revised form 23 July 2009
Accepted 27 July 2009
Available online 30 July 2009

PACS:
34.80.Dp

Keywords:
Atomic collision processes
Electron impact excitation
Electron impact ionisation
Intermediate energies
IERM
R-matrix
RMPS
Two-dimensional R-matrix propagation

ABSTRACT

The R-matrix method has proved to be a remarkably stable, robust and efficient technique for solving the close-coupling equations that arise in electron and photon collisions with atoms, ions and molecules. During the last thirty-four years a series of related R-matrix program packages have been published periodically in CPC. These packages are primarily concerned with low-energy scattering where the incident energy is insufficient to ionise the target. In this paper we describe 2DRMP, a suite of two-dimensional R-matrix propagation programs aimed at creating virtual experiments on high performance and grid architectures to enable the study of electron scattering from H-like atoms and ions at intermediate energies.

Program summary

Program title: 2DRMP
Catalogue identifier: AEEA_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEEA_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html
No. of lines in distributed program, including test data, etc.: 196 717
No. of bytes in distributed program, including test data, etc.: 3 819 727
Distribution format: tar.gz
Programming language: Fortran 95, MPI
Computer: Tested on CRAY XT4 [1]; IBM eServer 575 [2]; Itanium II cluster [3]
Operating system: Tested on UNICOS/lc [1]; IBM AIX [2]; Red Hat Linux Enterprise AS [3]
Has the code been vectorised or parallelised?: Yes. 16 cores were used for small test run
Classification: 2.4
External routines: BLAS, LAPACK, PBLAS, ScaLAPACK
Subprograms used: ADAZ_v1_1
Nature of problem: 2DRMP is a suite of programs aimed at creating virtual experiments on high performance architectures to enable the study of electron scattering from H-like atoms and ions at intermediate energies.
Solution method: Two-dimensional R-matrix propagation theory. The (r_1, r_2) space of the internal region is subdivided into a number of subregions. Local R-matrices are constructed within each subregion and used to propagate a global R-matrix, $\Re$, across the internal region. On the boundary of the internal region $\Re$ is transformed onto the IERM target state basis. Thus, the two-dimensional R-matrix propagation technique transforms an intractable problem into a series of tractable problems enabling the internal region to be extended far beyond that which is possible with the standard one-sector codes. A distinctive feature of the method is that both electrons are treated identically and the R-matrix basis states are constructed to allow for both electrons to be in the continuum. The subregion size is flexible and can be adjusted to accommodate the number of cores available.
Restrictions: The implementation is currently restricted to electron scattering from H-like atoms and ions.

DOI of original article: 10.1016/j.cpc.2009.07.017.
[☆] This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
[*] Corresponding author.
E-mail address: ns.scott@qub.ac.uk (N.S. Scott).

0010-4655/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2009.07.018

Additional comments: The programs have been designed to operate on serial computers and to exploit the distributed memory parallelism found on tightly coupled high performance clusters and supercomputers. 2DRMP has been systematically and comprehensively documented using ROBODoc [4] which is an API documentation tool that works by extracting specially formatted headers from the program source code and writing them to documentation files.
Running time: The wall clock running time for the small test run using 16 cores and performed on [3] is as follows: bp (7 s); rint2 (34 s); newrd (32 s); diag (21 s); amps (11 s); prop (24 s).
References:
[1] HECToR, CRAY XT4 running UNICOS/lc, http://www.hector.ac.uk/, accessed 22 July, 2009.
[2] HPCx, IBM eServer 575 running IBM AIX, http://www.hpcx.ac.uk/, accessed 22 July, 2009.
[3] HP Cluster, Itanium II cluster running Red Hat Linux Enterprise AS, Queen s University Belfast, http://www.qub.ac.uk/directorates/InformationServices/Research/HighPerformanceComputing/Services/Hardware/HPResearch/, accessed 22 July, 2009.
[4] Automating Software Documentation with ROBODoc, http://www.xs4all.nl/~rfsber/Robo/, accessed 22 July, 2009.

1. Introduction

R-matrix theory was first introduced in nuclear physics by Wigner [1,2]. Around the early 1970s it was realised that this approach could also be used in atomic and molecular physics [3]. Since then the R-matrix method has proved to be a remarkably stable, robust and efficient technique for solving the close-coupling equations that arise in electron and photon collisions with atoms, ions and molecules [4–6].

During the past thirty-five years a series of related R-matrix program packages have been published periodically in Computer Physics Communications (CPC). Building on the important foundational work of Allison [7], Burke [8], Hibbert [9] and Robb [10,11], Berrington et al. published in 1974 [12] and again in 1978 [13] an influential general program based on the non-relativistic Hamiltonian for calculating electron–atom and electron–ion cross sections as well as general atomic and photoionization cross sections and polarizabilities. This package was extended in 1982 by Scott and Taylor [14] to exploit model potentials and to allow for relativistic effects by including terms from the Breit–Pauli Hamiltonian. A non-exchange version of the non-relativistic package was subsequently published in 1992 by Burke, Burke and Scott [15]. In 1995, an updated version of the general package, to calculate electron–atom and electron–ion collision processes, with options to calculate radiative data and photoionization in either LS-coupling or in an intermediate-coupling scheme, was published by Berrington, Eissner and Norrington [16]. This program was based on two earlier CPC packages [13,14] and included extensions by the Opacity Project [17,18] and the Iron Project team [19].[1] More recently, in 2006, Zatsarinny [20] published a novel implementation of the R-matrix method with two significant innovations compared to the existing codes: non-orthogonal orbitals are used to represent both the bound and continuum one-electron orbitals; and a set of B-splines are used to define the R-matrix basis functions.

The aforementioned packages are primarily concerned with low-energy scattering where the incident energy is insufficient to ionize the target. At intermediate energies, from close to the ionization threshold to several times this energy, modelling of the scattering processes is difficult because account must be taken of the coupling amongst the infinite number of continuum states of the ionized target and the infinite number of target bound states lying below the ionization threshold. Two R-matrix approaches have been developed to represent this coupling with acceptable accuracy. The first is the R-matrix with Pseudostates Method (RMPS) [21–24]. Here, the standard R-matrix target eigenstate expansion is augmented by the inclusion of additional quadratically integrable pseudostates. These pseudostates are constructed by including additional contracted pseudo-orbitals in the target orbital basis. In this way the exact spectrum of the target is replaced by an approximate discrete spectrum. The advantage of the RMPS approach is that it is easy to implement in the standard R-matrix codes referenced in the paragraph above and is therefore readily applicable to general atomic or ionic targets. The second approach is the Intermediate Energy R-matrix Method (IERM) [25]. To date this method has only been implemented for one-electron targets [26] where the distinctive feature is that both electrons are treated identically and the R-matrix basis states are constructed to allow for both electrons to be in the continuum. The IERM approach results in a more densely packed pseudostate basis, with respect to the target state energy levels, than the RPMS basis. Accordingly it is more appropriate than RMPS in the study of scattering processes such as electron impact ionization close to the ionization threshold [27].

R-matrix theory is based around dividing the configuration space describing the collision process into two regions by a sphere of radius $r = a$, where a is chosen so that the charge distribution of the target atom or ion is contained within the sphere. Assuming a fixed range of incident scattering energies, it can be shown that number of IERM basis functions required within the internal region is roughly proportional to a^2 [28]. Accordingly, as a increases to allow the study of excitation to higher lying, and more diffuse, target states the calculation can rapidly become computationally intractable. Therefore an extension of IERM has been developed in which the (r_1, r_2) space of the internal region is subdivided into a number of subregions [25,29]. Local R-matrices are constructed within each subregion and used to propagate a global R-matrix, $\mathfrak{R}$, across the internal region. On the boundary between the internal and external regions $\mathfrak{R}$ is transformed onto the IERM target state basis. Thus, the two-dimensional R-matrix propagation technique transforms an intractable problem into a series of tractable problems enabling the internal region to be extended far beyond that which is possible with the standard one-sector codes: for example, in [30] the internal region is 600 a.u.

This paper describes 2DRMP, a suite of two-dimensional R-matrix propagation programs aimed at creating virtual experiments on high performance [31–35] and grid architectures [36,37] to enable the study of electron scattering from H-like atoms and ions at intermediate

[1] Unpublished variants of these codes are also available from http://amdpp.phys.strath.ac.uk/tamoc/code.html (accessed 22 July, 2009).

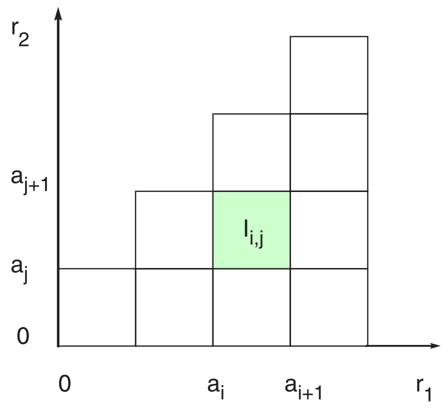


Fig. 1. Subdivision of the (r_1, r_2) plane within the internal region into a set of connected subregions $I_{i,j}$.

energies [27,30].[2] Two-dimensional R-matrix propagation theory is described in the following section, an outline of the 2DRMP software system is provided in Section 3, installation notes are given in Section 4 and concluding remarks are made in Section 5.

2. Two-dimensional R-matrix propagation theory

2.1. The R-matrix basis within a subregion

The 2-D R-matrix propagator method proceeds by subdividing the (r_1, r_2) plane within the R-matrix inner-region into a set of connected subregions, $I_{i,j}$, as shown in Fig. 1.

Within each subregion the wavefunction describing the two-electron system is expanded in an orthonormal set of energy independent basis functions,

$$\Psi_E(q_1, q_2) = \sum_{k=1}^{\infty} A_{Ek}\psi_k(q_1, q_2), \tag{1}$$

where E is total energy of the system and q_i represents the space, $\mathbf{r}_i$, and spin coordinates, σ_i, of electron i. The basis functions, $\psi_k(q_1, q_2)$, are eigenfunctions of the non-relativistic Schrödinger equation,

$$\mathcal{H}\psi_k(q_1, q_2) = E_k\psi_k(q_1, q_2), \tag{2}$$

where the Hamiltonian $\mathcal{H}$ is given in atomic units by,

$$\left(-\frac{1}{2}\nabla_1^2 - \frac{1}{2}\nabla_2^2 - \frac{Z}{r_1} - \frac{Z}{r_2} + \frac{1}{r_{12}}\right), \tag{3}$$

Z being the nuclear charge of the atom. Denoting the total orbital and spin angular momenta and their z components by L, S, M_L and M_S respectively, we note that since the Hamiltonian operator commutes with $\mathbf{L}^2$, L_z, $\mathbf{S}^2$, S_z and the parity Π, the basis functions are simultaneously eigenfunctions of these operators, i.e.

$$\left\langle \psi_k^{LM_LSM_S\Pi}(q_1, q_2)\middle|\mathcal{H}\middle|\psi_{k'}^{L'M_L'S'M_S'\Pi'}(q_1, q_2)\right\rangle = E_k\delta_{kk'}\delta_{LL'}\delta_{M_LM_L'}\delta_{SS'}\delta_{M_SM_S'}\delta_{\Pi\Pi'}. \tag{4}$$

Since the Hamiltonian is independent of electron spin each basis function is a product of a spatial and a spin function as follows:

$$\psi_k^{LM_LSM_S\Pi}(q_1, q_2) = \theta_k^{LM_LS\Pi}(\mathbf{r}_1, \mathbf{r}_2)\chi^{SM_S}(\sigma_1, \sigma_2). \tag{5}$$

The spin function, $\chi^{SM_S}(\sigma_1, \sigma_2)$, is defined in the normal way[3] and is antisymmetric when $S = 0$ and symmetric when $S = 1$. Eq. (4) therefore becomes,

$$\left\langle \theta_k^{LM_LS\Pi}(\mathbf{r}_1, \mathbf{r}_2)\middle|\mathcal{H}\middle|\theta_{k'}^{L'M_L'S'\Pi'}(\mathbf{r}_1, \mathbf{r}_2)\right\rangle = E_k\delta_{kk'}\delta_{LL'}\delta_{M_LM_L'}\delta_{SS'}\delta_{\Pi\Pi'}. \tag{7}$$

[2] This code is complementary to one developed by Dunseath et al. [38].

[3]
$$\chi^{0,0}(\sigma_1, \sigma_2) = \frac{1}{\sqrt{2}}\big(\alpha(1)\beta(2) - \beta(1)\alpha(2)\big), \tag{6a}$$
$$\chi^{1,1}(\sigma_1, \sigma_2) = \alpha(1)\beta(2), \tag{6b}$$
$$\chi^{1,0}(\sigma_1, \sigma_2) = \frac{1}{\sqrt{2}}\big(\alpha(1)\beta(2) + \beta(1)\alpha(2)\big), \tag{6c}$$
$$\chi^{1,-1}(\sigma_1, \sigma_2) = \beta(1)\beta(2), \tag{6d}$$
with α and β being the single electron spin functions [39].

For notational convenience we now drop the superscripts L, M_L, S and Π. The spatial basis functions, $\theta_k(\mathbf{r}_1, \mathbf{r}_2)$, are expanded in each subregion in terms of a set of orthonormal two-electron functions, $\phi_{n_1l_1n_2l_2}(\mathbf{r}_1, \mathbf{r}_2)$, as follows:

$$\theta_k(\mathbf{r}_1, \mathbf{r}_2) = \sum_{n_1l_1n_2l_2} \phi_{n_1l_1n_2l_2}(\mathbf{r}_1, \mathbf{r}_2)\alpha_{n_1l_1n_2l_2,k}. \tag{8}$$

The precise form of Eq. (8) depends on whether the subregion under consideration is diagonal, $I_{i,i}$ or off-diagonal, $I_{i,j}$, $i \neq j$.

In a diagonal subregion the Pauli Exclusion Principle requires that the basis wavefunctions, $\psi_k(q_1, q_2)$, must be antisymmetric. From Eqs. (5)–(6d) this is achieved when each of the two-electron functions, $\phi_{n_1l_1n_2l_2}(\mathbf{r}_1, \mathbf{r}_2)$, is symmetric when $S = 0$ and antisymmetric when $S = 1$. Accordingly, the two-electron functions, $\phi_{n_1l_1n_2l_2}(\mathbf{r}_1, \mathbf{r}_2)$, are given by,

$$\phi_{nlnl}(\mathbf{r}_1, \mathbf{r}_2) = \nu_{nl}(r_1)r_1^{-1}\nu_{nl}(r_2)r_2^{-1}\mathcal{Y}_{llLM_L}(\widehat{\mathbf{r}}_1, \widehat{\mathbf{r}}_2), \tag{9}$$

when the two electrons are equivalent ($n_1l_1 = n_2l_2$), with the proviso, imposed by the Pauli Exclusion Principle, that $L + S$ is even, and

$$\phi_{n_1l_1n_2l_2}(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{\sqrt{2}}\big(1 + (-1)^S P_{12}\big)\big\{\nu_{n_1l_1}(r_1)r_1^{-1}\nu_{n_2l_2}(r_2)r_2^{-1}\mathcal{Y}_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1, \widehat{\mathbf{r}}_2)\big\}, \tag{10}$$

when the two electrons are non-equivalent ($n_1l_1 \neq n_2l_2$), P_{12} being the spatial exchange operator. For non-equivalent electrons we note that,

$$\phi_{n_1l_1n_2l_2}(\mathbf{r}_1, \mathbf{r}_2) = \pm\phi_{n_2l_2n_1l_1}(\mathbf{r}_1, \mathbf{r}_2). \tag{11}$$

Therefore, to avoid linear independence problems, the sum over n_1l_1 and n_2l_2 in Eq. (8) only includes terms where $n_1l_1 \geqslant n_2l_2$.[4]

In an off-diagonal subregion each electron is localised to a different region of space and the Pauli Principle does not apply. Accordingly, the two-electron functions, $\phi_{n_1l_1n_2l_2}(\mathbf{r}_1, \mathbf{r}_2)$, are given by,

$$\phi_{n_1l_1n_2l_2}(\mathbf{r}_1, \mathbf{r}_2) = \nu_{n_1l_1}(r_1)r_1^{-1}\nu_{n_2l_2}(r_2)r_2^{-1}\mathcal{Y}_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1, \widehat{\mathbf{r}}_2), \tag{12}$$

with no restriction in the sum over n_1l_1 and n_2l_2 in Eq. (8).

The coupled angular functions, $\mathcal{Y}_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1, \widehat{\mathbf{r}}_2)$, are defined as,

$$\mathcal{Y}_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1, \widehat{\mathbf{r}}_2) = \sum_{m_1m_2} C(l_1l_2L; m_1m_2M_L)Y_{l_1m_1}(\widehat{\mathbf{r}}_1)Y_{l_2m_2}(\widehat{\mathbf{r}}_2), \tag{13}$$

where, $Y_{lm}(\widehat{\mathbf{r}})$, are spherical harmonics and, $C(l_1l_2L; m_1m_2M_L)$, are Clebsch–Gordan coefficients as defined by Rose [40]. The radial functions, $\nu_{nl}(r)$, are normalised eigenfunctions of the Schrödinger equation,

$$\left(\frac{d^2}{dr^2} - \frac{l(l+1)}{r^2} + \frac{2Z}{r} + k_{nl}^2\right)\nu_{nl}(r) = 0, \tag{14}$$

solved subject to the R-matrix boundary conditions,

$$\nu_{nl}(0) = 0, \tag{15a}$$

$$\frac{a_1}{\nu_{nl}(a_1)} \cdot \frac{d\nu_{nl}}{dr}\bigg|_{r=a_1} = 0, \tag{15b}$$

when $0 \leqslant r \leqslant a_1$ and

$$\frac{a_i}{\nu_{nl}(a_i)} \cdot \frac{d\nu_{nl}}{dr}\bigg|_{r=a_i} = 0, \tag{16a}$$

$$\frac{a_{i+1}}{\nu_{nl}(a_{i+1})} \cdot \frac{d\nu_{nl}}{dr}\bigg|_{r=a_{i+1}} = 0, \tag{16b}$$

when $a_i \leqslant r \leqslant a_{i+1}$, $i \geqslant 1$. These boundary conditions ensure that the Hamiltonian operator is Hermitian within the subregion.

2.1.1. Hamiltonian matrix elements within a subregion

The expansion coefficients in Eq. (8), $\alpha_{n_1l_1n_2l_2,k}$, are the normalised[5] eigenvectors obtained when the real symmetric matrix,

$$\big\langle\phi_{n_1l_1n_2l_2}(\mathbf{r}_1, \mathbf{r}_2)\big|\mathcal{H}\big|\phi_{n_3l_3n_4l_4}(\mathbf{r}_1, \mathbf{r}_2)\big\rangle, \tag{17}$$

is diagonalised: this ensures that Eq. (7) is satisfied. Again the form of this equation depends on whether the subregion is a diagonal or an off-diagonal subregion.

In a diagonal subregion there are four cases to consider: $n_1l_1 \neq n_2l_2$, $n_3l_3 \neq n_4l_4$; $n_1l_1 = n_2l_2$, $n_3l_3 = n_4l_4$; $n_1l_1 = n_2l_2$, $n_3l_3 \neq n_4l_4$; and $n_1l_1 \neq n_2l_2$, $n_3l_3 = n_4l_4$. These are given respectively as follows:

[4] I.e. $l_1 \geqslant l_2$ with the proviso that if $l_1 > l_2$ there are no restrictions on n_1 and n_2 but if $l_1 = l_2$ then $n_1 > n_2$ when $L + S$ is odd and $n_1 \geqslant n_2$ when $L + S$ is even.

[5] Normalised such that the sum of the squares of the moduli of the components are unity.

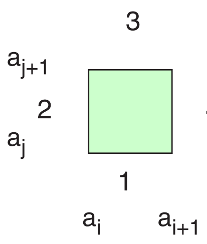


Fig. 2. The four edges, 1, 2, 3 and 4, of a general subregion.

$n_1l_1 \neq n_2l_2, n_3l_3 \neq n_4l_4$:

$$\begin{aligned}&\langle\phi_{n_1l_1n_2l_2}(\mathbf{r}_1,\mathbf{r}_2)|\mathcal{H}|\phi_{n_3l_3n_4l_4}(\mathbf{r}_1,\mathbf{r}_2)\rangle\\ &\quad=\frac{1}{2}\delta_{l_1l_3}\delta_{l_2l_4}\delta_{n_1n_3}\delta_{n_2n_4}\left(k^2_{n_3l_3}+k^2_{n_4l_4}\right)+\sum_\lambda f_\lambda(l_1l_2l_3l_4;L)R^D_\lambda(n_1l_1n_2l_2n_3l_3n_4l_4)\\ &\qquad+(-1)^{L+S+l_1+l_2}\left[\frac{1}{2}\delta_{l_1l_4}\delta_{l_2l_3}\delta_{n_1n_4}\delta_{n_2n_3}\left(k^2_{n_3l_3}+k^2_{n_4l_4}\right)+\sum_\lambda f_\lambda(l_1l_2l_4l_3;L)R^D_\lambda(n_1l_1n_2l_2n_4l_4n_3l_3)\right].\end{aligned}\tag{18}$$

$n_1l_1 = n_2l_2, n_3l_3 = n_4l_4$:

$$\langle\phi_{nlnl}(\mathbf{r}_1,\mathbf{r}_2)|\mathcal{H}|\phi_{nlnl}(\mathbf{r}_1,\mathbf{r}_2)\rangle=k^2_{n_l}+\sum_\lambda f_\lambda(llll;L)R^D_\lambda(nlnlnlnl).\tag{19}$$

$n_1l_1 = n_2l_2, n_3l_3 \neq n_4l_4$:

$$\langle\phi_{nlnl}(\mathbf{r}_1,\mathbf{r}_2)|\mathcal{H}|\phi_{n_3l_3n_4l_4}(\mathbf{r}_1,\mathbf{r}_2)\rangle=\sqrt{2}\left[\delta_{ll_3}\delta_{ll_4}\delta_{nn_3}\delta_{nn_4}k^2_{nl}+\sum_\lambda f_\lambda(lll_3l_4;L)R^D_\lambda(nlnln_3l_3n_4l_4)\right].\tag{20}$$

$n_1l_1 \neq n_2l_2, n_3l_3 = n_4l_4$:

$$\langle\phi_{n_1l_1n_2l_2}(\mathbf{r}_1,\mathbf{r}_2)|\mathcal{H}|\phi_{nlnl}(\mathbf{r}_1,\mathbf{r}_2)\rangle=\sqrt{2}\left[\delta_{ll_1}\delta_{ll_2}\delta_{nn_1}\delta_{nn_2}k^2_{nl}+\sum_\lambda f_\lambda(l_1l_2ll;L)R^D_\lambda(n_1l_1n_2l_2nlnl)\right].\tag{21}$$

In an off-diagonal subregion,

$$\begin{aligned}\langle\phi_{n_1l_1n_2l_2}(\mathbf{r}_1,\mathbf{r}_2)|\mathcal{H}|\phi_{n_3l_3n_4l_4}(\mathbf{r}_1,\mathbf{r}_2)\rangle&=\frac{1}{2}\delta_{l_1l_3}\delta_{l_2l_4}\delta_{n_1n_3}\delta_{n_2n_4}\left(k^2_{n_3l_3}+k^2_{n_4l_4}\right)\\&\quad+\sum_\lambda f_\lambda(l_1l_2l_3l_4;L)R^{OD}_\lambda(n_1l_1n_2l_2n_3l_3n_4l_4).\end{aligned}\tag{22}$$

In Eqs. (18)–(22) the angular integrals, $f_\lambda(l_1l_2l_3l_4;L)$, are given by,

$$\begin{aligned}f_\lambda(l_1l_2l_3l_4;L)&=(-1)^{l_1+l_3+L}\left[(2l_1+1)(2l_2+1)(2l_3+1)(2l_4+1)\right]^{1/2}\\&\quad\times(2\lambda+1)^{-1}C(l_1l_3\lambda;000)C(l_2l_4\lambda;000)W(l_1l_2l_3l_4;L\lambda),\end{aligned}\tag{23}$$

where, $W(l_1l_2l_3l_4;L\lambda)$, is a Racah coefficient as defined by Rose [40]. The two-dimensional radial integrals found on a diagonal subregion, $R^D_\lambda(n_1l_1n_2l_2n_3l_4n_3l_4)$, are given by,

$$R^D_\lambda(n_1l_1n_2l_2n_3l_4n_3l_4)=\int_{a_i}^{a_{i+1}}\int_{a_i}^{a_{i+1}}\nu_{n_1l_1}(r_1)\nu_{n_2l_2}(r_2)\frac{r^\lambda_<}{r^{\lambda+1}_>}\nu_{n_3l_3}(r_1)\nu_{n_4l_4}(r_2)\,dr_1\,dr_2,\tag{24}$$

whereas those found on an off-diagonal subregion, $R^{OD}_\lambda(n_1l_1n_2l_2n_3l_4n_3l_4)$, are given by,

$$R^{OD}_\lambda(n_1l_1n_2l_2n_3l_4n_3l_4)=\int_{a_i}^{a_{i+1}}\frac{\nu_{n_1l_1}(r_1)\nu_{n_3l_3}(r_1)}{r_1^{\lambda+1}}dr_1\int_{a_j}^{a_{j+1}}\nu_{n_2l_2}(r_2)\nu_{n_4l_4}(r_2)r_2^\lambda\,dr_2.\tag{25}$$

2.2. Local R-matrices within a subregion

Each general subregion in Fig. 1 has four edges labelled 1, 2, 3 and 4 as shown in Fig. 2. Within each general subregion the wavefunction on one edge is related to the first derivative of the wavefunction on all four edges by local R-matrices. In this section we derive these relationships.

In a diagonal subregion Eq. (8) can be expressed as,

$$\theta_k(\mathbf{r}_1,\mathbf{r}_2)=\left[\left(\sum_{l_1=0}^{l_{max}}\sum_{n_1=l_1+1}^{\infty}\sum_{l_2=0}^{l_1-1}\sum_{n_2=l_2+1}^{\infty}+\sum_{l_1=0}^{l_{max}}\sum_{n_1=l_1+1}^{\infty}\sum_{l_2=l_1}^{l_1}\sum_{n_2=l_2+1}^{n_1-1}\right)\right.$$

$$\left.\times\,\nu_{n_1l_1}(r_1)r_1^{-1}\nu_{n_2l_2}(r_2)r_2^{-1}\mathcal{Y}_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_2)\frac{\alpha_{n_1l_1n_2l_2,k}}{\sqrt{2}}\right] \tag{26a}$$

$$+\left[\left(\sum_{l_1=0}^{l_{max}}\sum_{n_1=l_1+1}^{\infty}\sum_{l_2=0}^{l_1-1}\sum_{n_2=l_2+1}^{\infty}+\sum_{l_1=0}^{l_{max}}\sum_{n_1=l_1+1}^{\infty}\sum_{l_2=l_1}^{l_1}\sum_{n_2=l_2+1}^{n_1-1}\right)\right.$$

$$\left.\times\,\nu_{n_2l_2}(r_1)r_1^{-1}\nu_{n_1l_1}(r_2)r_2^{-1}\mathcal{Y}_{l_2l_1LM_L}(\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_2)\frac{(-1)^{l_1+l_2+L+S}\alpha_{n_1l_1n_2l_2,k}}{\sqrt{2}}\right] \tag{26b}$$

$$+\sum_{l_1=0}^{l_{max}}\sum_{n_1=l_1+1}^{\infty}\nu_{n_1l_1}(r_1)r_1^{-1}\nu_{n_1l_1}(r_2)r_2^{-1}\mathcal{Y}_{l_1l_1LM_L}(\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_2)\alpha_{n_1l_1n_1l_1,k}. \tag{26c}$$

By relabelling $n_1l_1 \leftrightarrow n_2l_2$ in Eq. (26b), Eq. (8) becomes,

$$\theta_k(\mathbf{r}_1,\mathbf{r}_2)=\sum_{l_1=0}^{l_{max}}\sum_{n_1=l_1+1}^{\infty}\sum_{l_2=0}^{l_{max}}\sum_{n_2=l_2+1}^{\infty}\nu_{n_1l_1}(r_1)r_1^{-1}\nu_{n_2l_2}(r_2)r_2^{-1}\mathcal{Y}_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_2)a_{n_1l_1n_2l_2,k}, \tag{27}$$

where,

$$a_{n_1l_1n_2l_2,k}=\frac{1}{\sqrt{2}}\alpha_{n_1l_1n_2l_2,k} \qquad n_1l_1>n_2l_2, \tag{28a}$$

$$=\frac{1}{\sqrt{2}}(-1)^{L+S+l_1+l_2}\alpha_{n_2l_2n_1l_1,k} \quad n_2l_2>n_1l_1, \tag{28b}$$

$$a_{nlnl,k}=\alpha_{nlnl,k}\ \ \mathrm{mod}(L+S,2)=0. \tag{28c}$$

In an off-diagonal subregion, Eq. (8) also takes the form,

$$\theta_k(\mathbf{r}_1,\mathbf{r}_2)=\sum_{l_1=0}^{l_{max}}\sum_{n_1=l_1+1}^{\infty}\sum_{l_2=0}^{l_{max}}\sum_{n_2=l_2+1}^{\infty}\nu_{n_1l_1}(r_1)r_1^{-1}\nu_{n_2l_2}(r_2)r_2^{-1}\mathcal{Y}_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_2)a_{n_1l_1n_2l_2,k}, \tag{29}$$

but here,

$$a_{n_1l_1n_2l_2,k}=\alpha_{n_1l_1n_2l_2,k}. \tag{30}$$

In both diagonal and off-diagonal subregions terms that violate either of the following two conditions are excluded from the sum in Eq. (8),

$$|l_1-l_2|\leqslant L\leqslant l_1+l_2, \tag{31}$$

$$\mathrm{mod}(l_1+l_2,2)\neq\Pi. \tag{32}$$

We now determine the energy dependent coefficients A_{Ek} by considering,

$$\langle\psi_k|\mathcal{H}|\Psi_E\rangle-\langle\Psi_E|\mathcal{H}|\psi_k\rangle=(E-E_k)\langle\Psi_E|\psi_k\rangle, \tag{33}$$

where use has been made of Eq. (2) and we have assumed that Ψ_E, as defined in Eq. (1), satisfies,

$$\mathcal{H}\Psi_E=E\Psi_E. \tag{34}$$

Using Eqs. (1), (5), (27) (29) in Eq. (33) and defining the surface amplitudes as,

$$\omega_{n_2l_1l_2,k}(r_1)=\sum_{n_1=l_1+1}^{\infty}\nu_{n_1l_1}(r_1)a_{n_1l_1n_2l_2,k}, \tag{35}$$

$$\omega_{n_1l_1l_2,k}(r_2)=\sum_{n_2=l_2+1}^{\infty}\nu_{n_2l_2}(r_2)a_{n_1l_1n_2l_2,k}, \tag{36}$$

the radial wavefunction in channel $n_2l_1l_2$ and channel $n_1l_1l_2$ as,

$$y_{n_2l_1l_2}(r_1)=\sum_{k=1}^{\infty}A_{Ek}\omega_{n_2l_1l_2,k}(r_1), \tag{37}$$

$$y_{n_1l_1l_2}(r_2)=\sum_{k=1}^{\infty}A_{Ek}\omega_{n_1l_1l_2,k}(r_2), \tag{38}$$

and noting that the only non-zero contribution from the Hamiltonian, $\mathcal{H}$, is $(\frac{d^2}{dr_1^2}+\frac{d^2}{dr_2^2})$, it can be shown that,[6]

$$A_{Ek}=\frac{1}{2(E_k-E)}\sum_{nl_1l_2}\big(\omega_{(4)nl_1l_2,k}(a_{i+1})y'_{(4)nl_1l_2}(a_{i+1})-\omega_{(2)nl_1l_2,k}(a_i)y'_{(2)nl_1l_2}(a_i)$$
$$+\,\omega_{(3)nl_1l_2,k}(a_{j+1})y'_{(3)nl_1l_2}(a_{j+1})-\omega_{(1)nl_1l_2,k}(a_j)y'_{(1)nl_1l_2}(a_j)\big), \tag{39}$$

where, the surface amplitudes associated with of the four edges are defined as follows,

$$\omega_{(1)nl_1l_2,k}=\sum_{n'=l_2+1}^{\infty}a_{nl_1n'l_2,k}\nu_{n'l_2}(a_j), \tag{40}$$

$$\omega_{(2)nl_1l_2,k}=\sum_{n'=l_1+1}^{\infty}a_{n'l_1nl_2,k}\nu_{n'l_1}(a_i), \tag{41}$$

$$\omega_{(3)nl_1l_2,k}=\sum_{n'=l_2+1}^{\infty}a_{nl_1n'l_2,k}\nu_{n'l_2}(a_{j+1}), \tag{42}$$

$$\omega_{(4)nl_1l_2,k}=\sum_{n'=l_1+1}^{\infty}a_{n'l_1nl_2,k}\nu_{n'l_1}(a_{i+1}). \tag{43}$$

Substituting Eq. (39) into Eqs. (37) and (38) and evaluating on each of the four edges gives,

$$y_{(j)n'l'_1l'_2}=\sum_{i=1}^{4}\sum_{nl_1l_2}R_{(j,i)n'l'_1l'_2,nl_1l_2}y'_{(i)nl_1l_2}\eta_i, \quad j\in\{1,2,3,4\}, \tag{44}$$

where, $\eta_1=\eta_2=-1$ and $\eta_3=\eta_4=+1$. The local R-matrix, $R_{(j,i)n'l'_1l'_2,nl_1l_2}$, that relates channel $n'l'_1l'_2$ on edge j with channel nl_1l_2 on edge i is given by,

$$R_{(j,i)n'l'_1l'_2,nl_1l_2}=\frac{1}{2}\sum_{k=1}^{\infty}\frac{\omega_{(j)n'l'_1l'_2,k}\omega_{(i)nl_1l_2,k}}{E_k-E}, \quad j,i\in\{1,2,3,4\}. \tag{45}$$

Rewriting in a more convenient matrix notation we have,

$$\underline{y}_{(j)}=\sum_{i=1}^{4}\mathbf{R}_{(j,i)}\underline{y}'_{(i)}\eta_i, \quad j\in\{1,2,3,4\}. \tag{46}$$

2.2.1. The Buttle correction

In the preceding sections we have assumed that the sums over n_1, n_2 and k are infinite. In practice, n_1 and n_2 are bounded by n_{max} and k, the size of the Hamiltonian matrix, is bounded by k_{max}.[7] To ensure completeness in each channel the higher lying eigenstates in Eq. (1), $k>k_{max}$, are approximated by solutions of a zero-order potential scattering problem, with Hamiltonian, $-\frac{1}{2}\nabla^2-\frac{Z}{r}$, where each electron is treated independently.

Using an analysis similar to that in the preceding section it can be shown that the radial wavefunction in channel nl_1l_2 in the $\mathbf{r}_1$ direction is given by,

$$y_{nl_1l_2}(r_1)=\sum_{n'=l_1+1}^{\infty}\frac{\nu_{n'l_1}(r_1)}{(k^2_{n'l_1}-k^2)}\big[\nu_{n'l_1}(a_{i+1})y'_{nl_1l_2}(a_{i+1})-\nu_{n'l_1}(a_i)y'_{nl_1l_2}(a_i)\big], \tag{47}$$

where the channel energy, k^2 is given by,

$$k^2=2E-k^2_{nl_2}. \tag{48}$$

Here the radial functions, ν_{nl}, and their corresponding eigenvalues, k^2_{nl}, are solutions of Eqs. (14)–(16b). A similar analysis can be performed in the $\mathbf{r}_2$ direction.

Evaluating the radial wavefunctions in channel nl_1l_2 on the four edges now gives,

$$y_{(j)nl_1l_2}=\sum_{i}\mathcal{R}_{(j,i)nl_1l_2,nl_1l_2}y'_{(i)nl_1l_2}\eta_i, \quad i,j\in\{1,2,3,4\}, \tag{49}$$

where,

[6] It should be noted that for edges 1 and 3, $\sum_{nl_1l_2}=(\sum_{l_1=0}^{l_{max}}\sum_{l_2=0}^{l_{max}}\sum_{n=l_2+1}^{\infty})$, whereas for edges 2 and 4, $\sum_{nl_1l_2}=(\sum_{l_1=0}^{l_{max}}\sum_{n=l_1+1}^{\infty}\sum_{l_2=0}^{l_{max}})$.

[7] $k_{max}<(l_{max}+1)^2n^2_{max}$.

$$\mathcal{R}_{(j,i)nl_1l_2,nl_1l_2} = \sum_{n'=l_2+1}^{\infty} \frac{\nu_{(j)n'l_2}\nu_{(i)n'l_2}}{k^2_{n'l_2} - (2E - k^2_{nl_1})}, \quad j,i \in \{1,3\}, \tag{50a}$$

$$\mathcal{R}_{(j,i)nl_1l_2,nl_1l_2} = \sum_{n'=l_1+1}^{\infty} \frac{\nu_{(j)n'l_1}\nu_{(i)n'l_1}}{k^2_{n'l_1} - (2E - k^2_{nl_2})}, \quad j,i \in \{2,4\}, \tag{50b}$$

$$\mathcal{R}_{(j,i)nl_1l_2,nl_1l_2} = 0, \qquad |i-j| = 1 \text{ or } 3,\ j,i \in \{1,2,3,4\}. \tag{50c}$$

The sum in Eqs. (50a)–(50c), $\sum_{n=n_{max}+1}^{\infty}$, provides a correction for the truncated expansion, $\sum_{k=1}^{k_{max}}$, in Eq. (45). These correction matrices,

$$\mathcal{R}^c_{(j,i)nl_1l_2,nl_1l_2} = \sum_{n'=n_{max}+1}^{\infty} \frac{\nu_{(j)n'l_2}\nu_{(i)n'l_2}}{k^2_{n'l_2} - (2E - k^2_{nl_1})}, \quad j,i \in \{1,3\}, \tag{51a}$$

$$\mathcal{R}^c_{(j,i)nl_1l_2,nl_1l_2} = \sum_{n'=n_{max}+1}^{\infty} \frac{\nu_{(j)n'l_1}\nu_{(i)n'l_1}}{k^2_{n'l_1} - (2E - k^2_{nl_2})}, \quad j,i \in \{2,4\}, \tag{51b}$$

provide a correction to the diagonal elements of $R_{(1,1)}$, $R_{(1,3)}$, $R_{(3,1)}$, $R_{(3,3)}$ and $R_{(2,2)}$, $R_{(2,4)}$, $R_{(4,2)}$, $R_{(4,4)}$ respectively.

In the case of the off-diagonal correction matrices, $\mathcal{R}^c_{(1,3)}$, $\mathcal{R}^c_{(3,1)}$, $\mathcal{R}^c_{(2,4)}$ and $\mathcal{R}^c_{(4,2)}$, the sum is an alternating series. We have found through experiments with series acceleration that the correction is small and can be satisfactorily approximated by,

$$\mathcal{R}^c_{(j,i)nl_1l_2,nl_1l_2} \approx \frac{1}{2}\sum_{n'=n_{max}+1}^{n_{max}+10} \frac{\nu_{(j)n'l_2}\nu_{(i)n'l_2}}{k^2_{n'l_2} - (2E - k^2_{nl_1})}, \quad j,i \in \{1,3\},\ i \neq j, \tag{52a}$$

$$\mathcal{R}^c_{(j,i)nl_1l_2,nl_1l_2} \approx \frac{1}{2}\sum_{n'=n_{max}+1}^{n_{max}+10} \frac{\nu_{(j)n'l_1}\nu_{(i)n'l_1}}{k^2_{n'l_1} - (2E - k^2_{nl_2})}, \quad j,i \in \{2,4\},\ i \neq j. \tag{52b}$$

In the case of the diagonal correction matrices $\mathcal{R}^c_{(1,1)}$, $\mathcal{R}^c_{(2,2)}$, $\mathcal{R}^c_{(3,3)}$ and $\mathcal{R}^c_{(4,4)}$ the correction is much larger. Here we use a procedure similar to that described by Buttle [41]. Consider, for example, Eq. (47) evaluated on edge 3,

$$y_{(3)n_1l_1l_2} = \left(\sum_{n_2=l_2+1}^{\infty} \frac{\nu^2_{(3)n_2l_2}}{k^2_{n_2l_2} - k^2}\right) y'_{(3)n_1l_1l_2} - \left(\sum_{n_2=l_2+1}^{\infty} \frac{\nu_{(3)n_2l_2}\nu_{(1)n_2l_2}}{k^2_{n_2l_2} - k^2}\right) y'_{(1)n_1l_1l_2}. \tag{53}$$

Since, $y_{(i)nl_1l_2}$, is the solution of Eq. (14)[8] at the corresponding channel energy, k^2, provided the subregion is not too large, we find to a good approximation,

$$-y'_{(1)nl_1l_2} \approx +y'_{(3)nl_1l_2}, \tag{54}$$

$$\mathcal{R}_{(3,3)nl_1l_2,nl_1l_2} \gg \mathcal{R}_{(3,1)nl_1l_2,nl_1l_2}. \tag{55}$$

Eq. (53) thus becomes,

$$y_{(3)n_1l_1l_2} \approx \left(\sum_{n_2=l_2+1}^{\infty} \frac{\nu^2_{(3)n_2l_2}}{k^2_{n_2l_2} - k^2}\right) y'_{(3)n_1l_1l_2}. \tag{56}$$

Using the same approach on all edges in Eq. (49), the diagonal correction matrices can be approximated by,

$$\mathcal{R}^c_{(i,i)nl_1l_2,nl_1l_2} \approx \frac{y_{(i)nl_1l_2}}{y'_{(i)nl_1l_2}} - \sum_{n'=l_2+1}^{n_{max}} \frac{\nu_{(i)n'l_2}\nu_{(i)n'l_2}}{k^2_{n'l_2} - (2E - k^2_{nl_1})}, \quad i \in \{1,3\}, \tag{57a}$$

$$\mathcal{R}^c_{(i,i)nl_1l_2,nl_1l_2} \approx \frac{y_{(i)nl_1l_2}}{y'_{(i)nl_1l_2}} - \sum_{n'=l_1+1}^{n_{max}} \frac{\nu_{(i)n'l_1}\nu_{(i)n'l_1}}{k^2_{n'l_1} - (2E - k^2_{nl_2})}, \quad i \in \{2,4\}. \tag{57b}$$

2.3. Propagation of the global R-matrix across a general subregion

We now turn to the propagation of the R-matrix across a subregion [29,38]. Consider the general situation in Fig. 3 where we assume that we already know the global R-matrix, $\Re^I$, associated with the boundary defined by edges 5, 2, 1 and 6 in domain D and we wish to evaluate the new global R-matrix, $\Re^O$, associated with edges 5, 3, 4 and 6 in domain D' following propagation across subregion d. We note that because of symmetry we need only consider the lower half of domains D and D'.

We first rewrite Eq. (46) in subregion d as,

$$\begin{pmatrix} \underline{y}_I \\ \underline{y}_O \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_{II} & \mathbf{r}_{IO} \\ -\mathbf{r}_{OI} & \mathbf{r}_{OO} \end{pmatrix} \begin{pmatrix} \underline{y}'_I \\ \underline{y}'_O \end{pmatrix}, \tag{58}$$

[8] The solution on edge 3 is found by integrating Eq. (14) in the +ve direction, while the solution on edge 1 is found by integrating Eq. (14) in the negative direction.

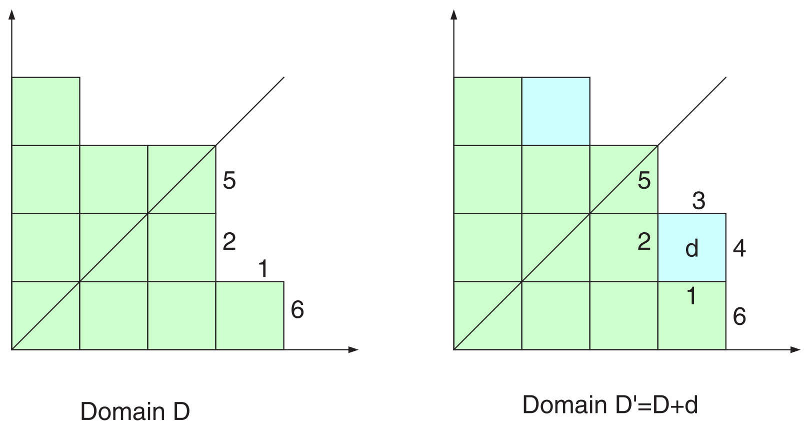


Fig. 3. Propagation of the R-matrix from domain D to domain D'.

where I represents the input edges 1 and 2, and O represents the output edges 3 and 4 so that,

$$\mathbf{r}_{II} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix}, \tag{59a}$$

$$\mathbf{r}_{IO} = \begin{pmatrix} \mathbf{R}_{13} & \mathbf{R}_{14} \\ \mathbf{R}_{23} & \mathbf{R}_{24} \end{pmatrix}, \tag{59b}$$

$$\mathbf{r}_{OI} = \begin{pmatrix} \mathbf{R}_{31} & \mathbf{R}_{32} \\ \mathbf{R}_{41} & \mathbf{R}_{42} \end{pmatrix}, \tag{59c}$$

$$\mathbf{r}_{OO} = \begin{pmatrix} \mathbf{R}_{33} & \mathbf{R}_{34} \\ \mathbf{R}_{43} & \mathbf{R}_{44} \end{pmatrix}. \tag{59d}$$

We can write the global R-matrix, $\Re^I$, in domain D as,

$$\Re^I = \begin{pmatrix} \Re^I_{II} & \Re^I_{IX} \\ \Re^I_{XI} & \Re^I_{XX} \end{pmatrix}, \tag{60}$$

where,

$$\begin{pmatrix} \underline{y}_I \\ \underline{y}_X \end{pmatrix} = \begin{pmatrix} \Re^I_{II} & \Re^I_{IX} \\ \Re^I_{XI} & \Re^I_{XX} \end{pmatrix} \begin{pmatrix} \underline{y}'_I \\ \underline{y}'_X \end{pmatrix} \tag{61}$$

with X denoting edges 5 and 6: these edges are common to domains D and D'. The global output R-matrix, $\Re^O$, in domain D' can be written as,

$$\Re^O = \begin{pmatrix} \Re^O_{OO} & \Re^O_{OX} \\ \Re^O_{XO} & \Re^O_{XX} \end{pmatrix}, \tag{62}$$

where,

$$\begin{pmatrix} \underline{y}_O \\ \underline{y}_X \end{pmatrix} = \begin{pmatrix} \Re^O_{OO} & \Re^O_{OX} \\ \Re^O_{XO} & \Re^O_{XX} \end{pmatrix} \begin{pmatrix} \underline{y}'_O \\ \underline{y}'_X \end{pmatrix}. \tag{63}$$

Manipulation of Eqs. (58), (61) and (63) results in the following expressions for $\Re^O$ in terms of $\Re^I$ and the local matrices $\mathbf{r}$,

$$\Re^O_{OO} = \mathbf{r}_{OO} - \mathbf{r}_{OI}\left(\mathbf{r}_{II} + \Re^I_{II}\right)^{-1}\mathbf{r}_{IO}, \tag{64a}$$

$$\Re^O_{OX} = \mathbf{r}_{OI}\left(\mathbf{r}_{II} + \Re^I_{II}\right)^{-1}\Re^I_{IX}, \tag{64b}$$

$$\Re^O_{XO} = \Re^I_{XI}\left(\mathbf{r}_{II} + \Re^I_{II}\right)^{-1}\mathbf{r}_{IO}, \tag{64c}$$

$$\Re^O_{XX} = \Re^I_{XX} - \Re^I_{XI}\left(\mathbf{r}_{II} + \Re^I_{II}\right)^{-1}\Re^I_{IX}. \tag{64d}$$

While Eqs. (64a)–(64c) can be applied to the propagation across a general subregion two special situations should noted: propagation across a diagonal subregion and propagation across a subregion bounded by the r_1-axis at the beginning of a new strip.

In the case of a diagonal subregion from symmetry considerations edge 2 is identical to edge 1 and edge 3 is identical to edge 4. Accordingly, with only one input edge and one output edge Eqs. (59a)–(59d) become,

$$\mathbf{r}_{II} = 2\mathbf{R}_{11}, \tag{65a}$$

$$\mathbf{r}_{IO} = 2\mathbf{R}_{14}, \tag{65b}$$

$$\mathbf{r}_{OI} = 2\mathbf{R}_{41}, \tag{65c}$$

$$\mathbf{r}_{OO} = 2\mathbf{R}_{44}. \tag{65d}$$

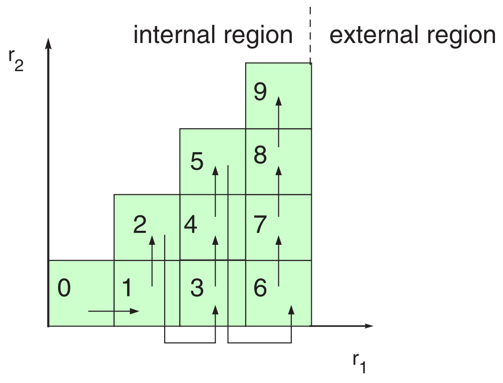


Fig. 4. Propagation of the R-matrix across the inner region.

In the case of a subregion bounded by the r_1-axis at the beginning of a new strip we note that the input boundary I consists of only one edge. When propagating across the first subregion in the second strip there is no common boundary X: in this case only Eq. (64a) need be solved.

Having obtained $\Re$ on the boundary of the innermost subregion (labelled 0 in Fig. 4), $\Re$ is propagated across each subregion in the order indicated, working systematically from the r_1-axis at the bottom of each strip across all subregions to the diagonal, eventually yielding the global R-matrix, $\Re$, on the boundary of the R-matrix internal and external regions.[9]

Before matching to the solution in the external region we first project the basis functions which span the elementary edges of this final boundary, onto the atomic electron basis used in the asymptotic region and transform the global R-matrix accordingly. This is described in the following section.

2.4. Transformation of the global R-matrix on the boundary between the internal and external regions

In the outer region, $r_1 \geqslant r_a$, electron exchange between the target electron and the scattered electron is ignored. Here the two-electron wavefunction is given by,

$$\Psi_E(\mathbf{r}_1,\mathbf{r}_2)=\sum_{n_2l_1l_2}\frac{y_{n_2l_1l_2}(r_1)}{r_1}\mathcal{Y}_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_2)\frac{P_{n_2l_2}(r_2)}{r_2},\quad r_1\geqslant r_a \tag{66}$$

where, $y_{n_2l_1l_2}(r_1)$ is the unknown radial function of the scattered electron in channel, $n_2l_1l_2$, and, $P_{n_2l_2}$, is the atomic electron basis. The atomic basis is finite such that $0 \leqslant l_2 \leqslant lbndmax$ and $l_2+1 \leqslant n_2 \leqslant nbndmax$.[10]

The boundary radius, r_a, is chosen so that the physical, or true, hydrogenic target orbitals of interest are accurately contained within this boundary. These orbitals are augmented with numerical pseudo-orbitals constructed to be solutions of,

$$\left(\frac{d^2}{dr_2^2}-\frac{l_2(l_2+1)}{r_2^2}+\frac{2Z}{r_2}+k^2_{n_2l_2}\right)P_{n_2l_2}(r_2)=0, \tag{67}$$

subject to the R-matrix boundary conditions,

$$P_{n_2l_2}(0)=0, \tag{68a}$$

$$\frac{r_a}{P_{n_2l_2}(r_a)}\cdot\frac{dP_{n_2l_2}}{dr}\bigg|_{r_2=r_a}=0. \tag{68b}$$

Provided the boundary is chosen large enough the physical orbitals and the pseudo-orbitals will be automatically orthogonal.

Using Eq. (66) the scattered electron's radial wavefunction in channel, $n_2l_1l_2$, can be written on the boundary, $r_1=r_a$, as,

$$\frac{y_{n_2l_1l_2}(r_a)}{r_a}=\int\frac{P_{n_2l_2}(r_2)}{r_2}\mathcal{Y}^*_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_2)\Psi_E(\mathbf{r}_1,\mathbf{r}_2)\,d\widehat{r}_1\,d\mathbf{r}_2,\quad r_1=r_a. \tag{69}$$

Following the propagation across the inner-region we have a contribution to Ψ_E on the boundary, $r_1=r_a$, from each subregion i. Denoting this contribution by, Ψ_E^i, we have,

$$\Psi_E^i(\mathbf{r}_1,\mathbf{r}_2)=\sum_{n_2l_1l_2}\frac{y^i_{n_2l_1l_2}(r_a)}{r_a}\mathcal{Y}_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_2)\frac{v^i_{n_2l_2}(r_2)}{r_2},\quad a_{i-1}\leqslant r_2\leqslant a_i. \tag{70}$$

Thus at, $r_1=r_a$, in subregion i, the contribution to the radial wavefunction in channel, $n_2l_1l_2$, is given by,

$$\frac{y_{n_2l_1l_2}(r_a)}{r_a}=\int\frac{P_{n_2l_2}(r_2)}{r_2}\mathcal{Y}^*_{l_1l_2LM_L}(\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_2)\Psi^i_E(\mathbf{r}_1,\mathbf{r}_2)\,d\widehat{r}_1\,d\mathbf{r}_2,\quad a_{i-1}\leqslant r_2\leqslant a_i. \tag{71}$$

[9] The route through the inner-region is not unique. For example, it is possible to proceed horizontally across rows rather than vertically in columns as illustrated in Fig. 4. However, the amount of computation required for each route is different. An informal proof that the route indicated in Fig. 4 is optimal is given in [33].

[10] The physical states are known exactly and may be represented as analytic functions of the form, $P_{nl}(r)=\sum_{i=1,nt} coef[i]*r^{irad[i]}*e^{-alpha*r}$, with $0\leqslant l\leqslant \min(lbndmax, nbound-1)$ and $l+1\leqslant n\leqslant nbound$. See Section 4.2 for information on the choice of `lbndmax`, `nbndmax` and `nbound`.

Substituting Eq. (70) into Eq. (71), and considering all subregions, we find that,

$$y_{n_2 l_1 l_2}(r_a) = \sum_{i=1}^{n_{strips}} \sum_{n=l_2+1}^{n_{max}} A^i_{n_2 l_2 n} y^i_{n l_1 l_2}(r_a), \tag{72}$$

where,

$$A^i_{n_2 l_2 n} = \int_{a_{i-1}}^{a_i} P_{n_2 l_2}(r_2) \nu^i_{n l_2}(r_2)\, dr_2. \tag{73}$$

For convenience we introduce a new notation for the radial basis functions, $\nu^i_{nl_2}$, and the radial wavefunction in each channel as follows,

$$\begin{aligned}
&\mu_{nl_2}(r_2) = \nu^1_{(n+l_2)l_2}(r_2), && n = 1 \ldots g,\\
&\mu_{(n+g)l_2}(r_2) = \nu^2_{(n+l_2)l_2}(r_2), && n = 1 \ldots g,\\
&\mu_{(n+2g)l_2}(r_2) = \nu^3_{(n+l_2)l_2}(r_2), && n = 1 \ldots g,\\
&\vdots\\
&\mu_{(n+(n_{strips}-1)g)l_2}(r_2) = \nu^{n_{strips}}_{(n+l_2)l_2}(r_2), && n = 1 \ldots g,
\end{aligned} \tag{74}$$

$$\begin{aligned}
&u_{nl_1l_2}(r_a) = y^1_{(n+l_2)l_1l_2}(r_a), && n = 1 \ldots g,\\
&u_{(n+g)l_1l_2}(r_a) = y^2_{(n+l_2)l_1l_2}(r_a), && n = 1 \ldots g,\\
&u_{(n+2g)l_1l_2}(r_a) = y^3_{(n+l_2)l_1l_2}(r_a), && n = 1 \ldots g,\\
&\vdots\\
&u_{(n+(n_{strips}-1)g)l_1l_2}(r_a) = y^{n_{strips}}_{(n+l_2)l_1l_2}(r_a), && n = 1 \ldots g,
\end{aligned} \tag{75}$$

where, $g = n_{max} - l_2$. Accordingly, Eq. (72) becomes,

$$y_{n_2 l_1 l_2}(r_a) = \sum_{i=1}^{g \times n_{strips}} B_{n_2 l_2 i} u_{i l_1 l_2}(r_a), \tag{76}$$

with,

$$B_{n_2 l_2 i} = \int_0^{g \times n_{strips}} P_{n_2 l_2}(r_2) \mu_{i l_2}(r_2)\, dr_2. \tag{77}$$

We can further introduce a matrix,

$$C_{n_2 l_1 l_2, i l'_1 l'_2} = B_{n_2 l_2 i} \delta_{l_1 l'_1} \delta_{l_2 l'_2}, \tag{78}$$

with the property, $\mathbf{C}^T \mathbf{C} = \mathbf{I}$, which relates the physical wavefunction, $y_{n_2 l_1 l_2}(r_a)$, on the boundary to the corresponding basis wavefunctions in the subregions, i.e.

$$\underline{y} = \mathbf{C} \underline{u}_O, \tag{79}$$

$$\underline{y}' = \mathbf{C} \underline{u}'_O. \tag{80}$$

Since,

$$\underline{u}_O = \Re^O_{OO} \underline{u}'_O, \tag{81}$$

and

$$\underline{y} = \mathbf{C} \Re^O_{OO} \mathbf{C}^T \underline{y}', \tag{82}$$

we find that the physical R-matrix on the boundary is related to the global R-matrix on the boundary through the unitary transformation,

$$\mathbf{R} = \mathbf{C} \Re^O_{OO} \mathbf{C}^T. \tag{83}$$

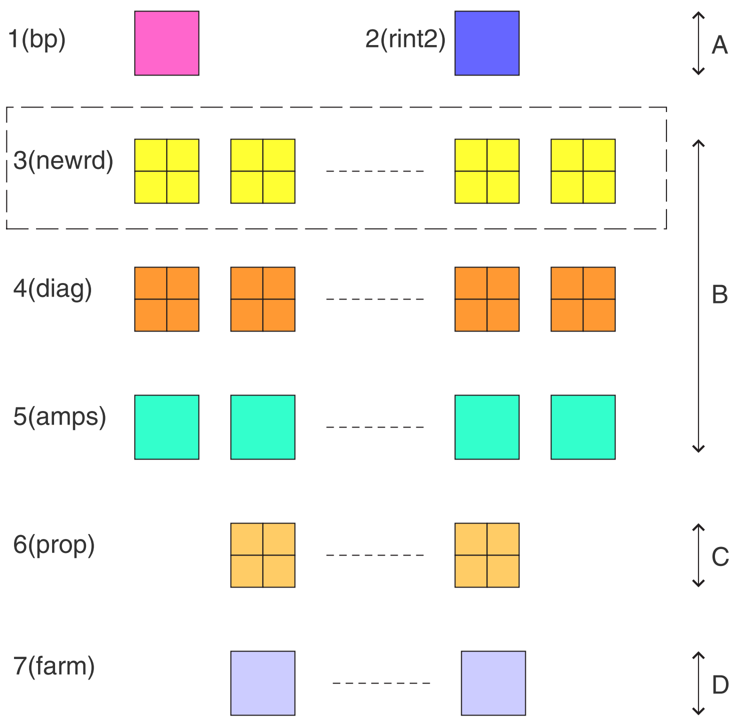


Fig. 5. The 2DRMP suit. Each program belongs to one of four functional blocks: A, B, C or D. The blocks must be performed sequentially and communication between programs is through files. Blocks A and B are independent of the collision energy and need only be performed once while blocks C and D are dependent on the collision energy and must be repeated hundreds or thousands of times.

2.5. The external region and the potential matrix V_{ij}

Finally, we briefly comment on the solution of the equations in the external region. In this region the equations are of the form found in the standard R-matrix method [43],

$$\left(\frac{d^2}{dr^2} - \frac{l_i(l_i+1)}{r^2} + k_i^2\right)\mathcal{F}_{ij}(r) = 2\sum_{p=1}^{N} V_{ip}\mathcal{F}_{pj}, \quad i, j = 1\ldots N. \tag{84}$$

In this equation, the channel orbital angular momenta are denoted by l_i, the channel momenta, k_i, by

$$k_i^2 = 2\left(E - E_i^T\right), \tag{85}$$

where E^T is the energy of the target corresponding to channel i and E is the incident energy. The interaction between the projectile and the target, V_{ij} is given by the electrostatic multipole expansion (see Eq. (7.33) of [26]),

$$V_{ij}(r) = \sum_{\lambda=1}^{\lambda_{max}} \frac{a_{ij}^{\lambda}}{r^{\lambda+1}}. \tag{86}$$

In Eq. (86) the long-range potential coefficients, a_{ij}^{λ}, simplify to (see Eqs. (7.34) and (7.35) of [26]),

$$a_{ij}^{\lambda} = f_{\lambda}(l_1, l_2, l_3, l_4 : L)I_{ij}^{\lambda}, \tag{87}$$

where,

$$I_{ij}^{\lambda} = \int_0^{r_a} P_{n_1 l_1}(r) r^{\lambda} P_{n_3 l_3}(r)\, dr, \tag{88}$$

and where $P_{n_1 l_1}(r)$ and $P_{n_3 l_3}$ refer to the atomic orbitals in channel i and j respectively.

Eqs. (84) are integrated outwards, subject to the R-matrix boundary conditions, using the FARM_2DRMP package [42], a modified version of FARM [43], to yield cross sections and other scattering observables of interest.

3. The 2DRMP software package outline

The two-dimensional propagator model described in the previous section has been implemented as a suite of seven programs, named 2DRMP, as depicted in Fig. 5. These programs have been designed to operate on serial computers and to exploit the distributed memory parallelism found on tightly coupled high performance clusters and supercomputers.

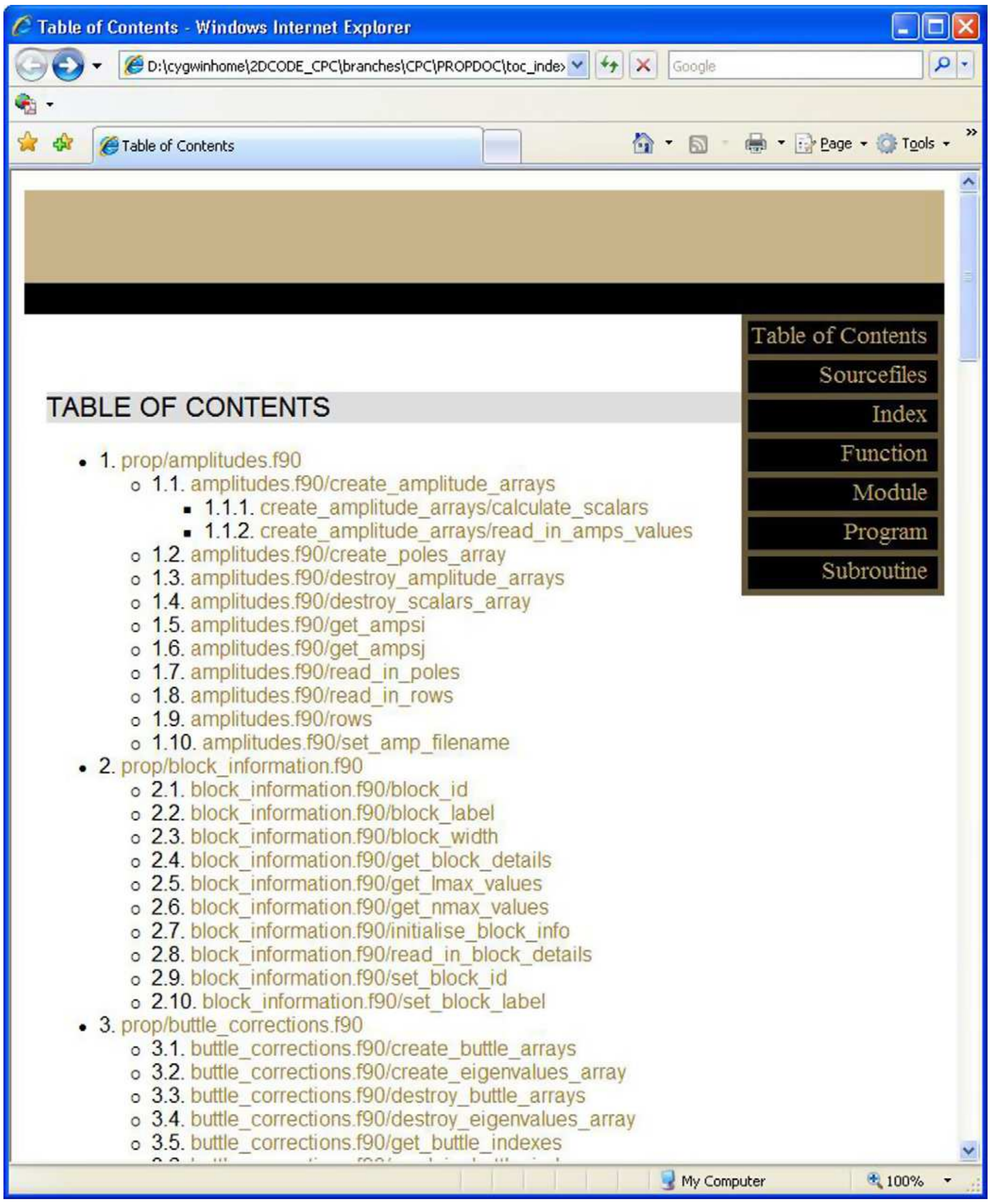


TABLE OF CONTENTS

- 1. prop/amplitudes.f90
  - 1.1. amplitudes.f90/create_amplitude_arrays
    - 1.1.1. create_amplitude_arrays/calculate_scalars
    - 1.1.2. create_amplitude_arrays/read_in_amps_values
  - 1.2. amplitudes.f90/create_poles_array
  - 1.3. amplitudes.f90/destroy_amplitude_arrays
  - 1.4. amplitudes.f90/destroy_scalars_array
  - 1.5. amplitudes.f90/get_ampsi
  - 1.6. amplitudes.f90/get_ampsj
  - 1.7. amplitudes.f90/read_in_poles
  - 1.8. amplitudes.f90/read_in_rows
  - 1.9. amplitudes.f90/rows
  - 1.10. amplitudes.f90/set_amp_filename
- 2. prop/block_information.f90
  - 2.1. block_information.f90/block_id
  - 2.2. block_information.f90/block_label
  - 2.3. block_information.f90/block_width
  - 2.4. block_information.f90/get_block_details
  - 2.5. block_information.f90/get_lmax_values
  - 2.6. block_information.f90/get_nmax_values
  - 2.7. block_information.f90/initialise_block_info
  - 2.8. block_information.f90/read_in_block_details
  - 2.9. block_information.f90/set_block_id
  - 2.10. block_information.f90/set_block_label
- 3. prop/buttle_corrections.f90
  - 3.1. buttle_corrections.f90/create_buttle_arrays
  - 3.2. buttle_corrections.f90/create_eigenvalues_array
  - 3.3. buttle_corrections.f90/destroy_buttle_arrays
  - 3.4. buttle_corrections.f90/destroy_eigenvalues_array
  - 3.5. buttle_corrections.f90/get_buttle_indexes

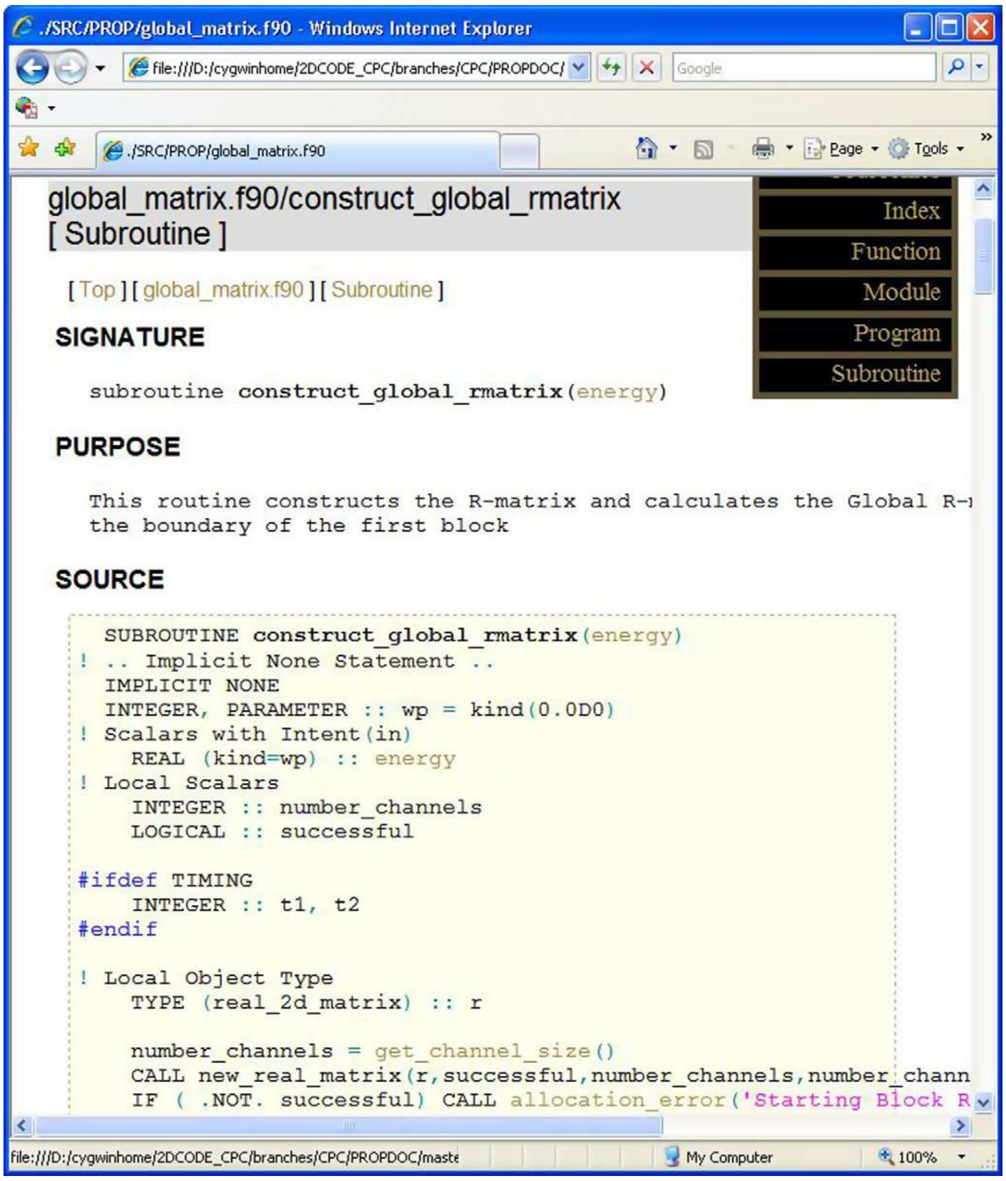
Fig. 6. The ROBODoc master index for Program 6 (prop). Separate indexes are provided for programs, modules, functions and subroutines.

2DRMP has been systematically and comprehensively documented using ROBODoc [44] which is an API documentation tool that works by extracting specially formatted headers from the program source code and writing them to documentation files. This allows a program and its documentation to be maintained in a single file. Each of the programs in Fig. 5 is accompanied by HTML documentation that can be accessed via a master index as illustrated in Fig. 6 for Program 6 (prop). Selecting one of the hyperlinks displays the corresponding documentation which comprises the routine's signature, purpose and source code. This is shown in Fig. 7 for `construct_global_rmatrix` which is an internal subprogram of module `global_rmatrix`.

We begin by sketching the architecture of the 2DRMP suit, highlighting the principal function of each program. Each program belongs to one of four functional blocks: A, B, C or D. These blocks must be completed sequentially. Blocks A and B are independent of the collision energy and need only be performed once while blocks C and D are dependent on the collision energy and must be repeated hundreds or thousands of times. Communication between programs is through files as illustrated in Fig. 8.

Block A contains two independent programs that are not computationally intensive. Program 1 (bp) constructs the atomic basis functions, defined by Eqs. (67)–(68b), and the long-range potential coefficients, defined by Eqs. (87)–(88), that are to be used to define the atomic target in the external region. Program 2 (rint2) computes the radial integrals, defined by Eq. (25), to be used in the construction of the Hamiltonian matrix in off-diagonal subregions.

In Block B: Program 3 (newrd) constructs a subregion Hamiltonian matrix, as defined by Eqs. (18)–(22); Program 4 (diag) diagonalises the corresponding matrix; and Program 5 (amps) constructs the surface amplitudes on the subregion's edges using the matrix's eigenvectors as defined by Eqs. (40)–(43). Each column in Block B corresponds to an independent subregion. The rows of Block B are pleasingly parallel. For example, Program 3 (newrd) can be used to construct a group of $n_{subregion}$ subregion matrices using n_{cps} cores for each


./SRC/PROP/global_matrix.f90 - Windows Internet Explorer
file:///D:/cygwinhome/2DCODE_CPC/branches/CPC/PROPDOC/
Google
./SRC/PROP/global_matrix.f90
Page
Tools
Index
Function
Module
Program
Subroutine
file:///D:/cygwinhome/2DCODE_CPC/branches/CPC/PROPDOC/maste
My Computer
100%

global_matrix.f90/construct_global_rmatrix
[Subroutine]

[Top] [global_matrix.f90] [Subroutine]

SIGNATURE

```
subroutine construct_global_rmatrix(energy)
```

PURPOSE

```
This routine constructs the R-matrix and calculates the Global R-
the boundary of the first block
```

SOURCE

```
   SUBROUTINE construct_global_rmatrix(energy)
!  .. Implicit None Statement ..
   IMPLICIT NONE
   INTEGER, PARAMETER :: wp = kind(0.0D0)
!  Scalars with Intent(in)
     REAL (kind=wp) :: energy
!  Local Scalars
     INTEGER :: number_channels
     LOGICAL :: successful

#ifdef TIMING
     INTEGER :: t1, t2
#endif

!  Local Object Type
     TYPE (real_2d_matrix) :: r

     number_channels = get_channel_size()
     CALL new_real_matrix(r,successful,number_channels,number_chann
     IF ( .NOT. successful) CALL allocation_error('Starting Block R
```

Fig. 7. ROBODoc documentation for `construct_global_rmatrix` comprising the subroutine's signature, purpose and source code: `construct_global_rmatrix` is an internal subprogram of module `global_rmatrix`.

subregion.[11] This is illustrated by the dashed line rectangle in Fig. 5 where each matrix construction is spread across a 2×2 grid of cores. Programs 4 (diag) and 5 (amps) have a similar capability, however, Program 5 (amps) has only been implemented for one core per subregion i.e. $n_{cps} = 1$.

Block C uses Program 6 (prop) to propagate the global R-matrix, $\Re$, across all the subregions of the inner-region, in the order dictated by Fig. 4, using Eqs. (64a)–(64d). Each element in this block corresponds to a series of propagations across a range of scattering energies. The k elements can be computed in parallel with n_{cpe} cores devoted to each element.[12] When the boundary between the internal and external region is reached the global R-matrix is transformed through the unitary transformation described by Eq. (83).

The final block, Block D, corresponds to the solution of Eq. (84). Again each element in this block corresponds to a range of scattering energies. In the source code distribution we have included a modified version of FARM [43], FARM_2DRMP [42] that allows k elements to be computed simultaneously across k cores.

We now turn to a description of the salient features of each of the programs.

3.1. Program 1 (bp)

Program 1 (bp) constructs the atomic basis functions, defined by Eqs. (67)–(68b), and the long-range potential coefficients, defined by Eqs. (87)–(88). These are used to define the atomic target in the external region. The program's call graph is shown below and is followed by a brief description of the key subroutines.

[11] $n_{subregion} \geqslant 1$, $n_{cps} \geqslant 1$ and $n_{core} = n_{subregion} \times n_{cps}$.
[12] $k \geqslant 1$, $n_{cpe} \geqslant 1$ and $n_{core} = k \times n_{cpe}$.

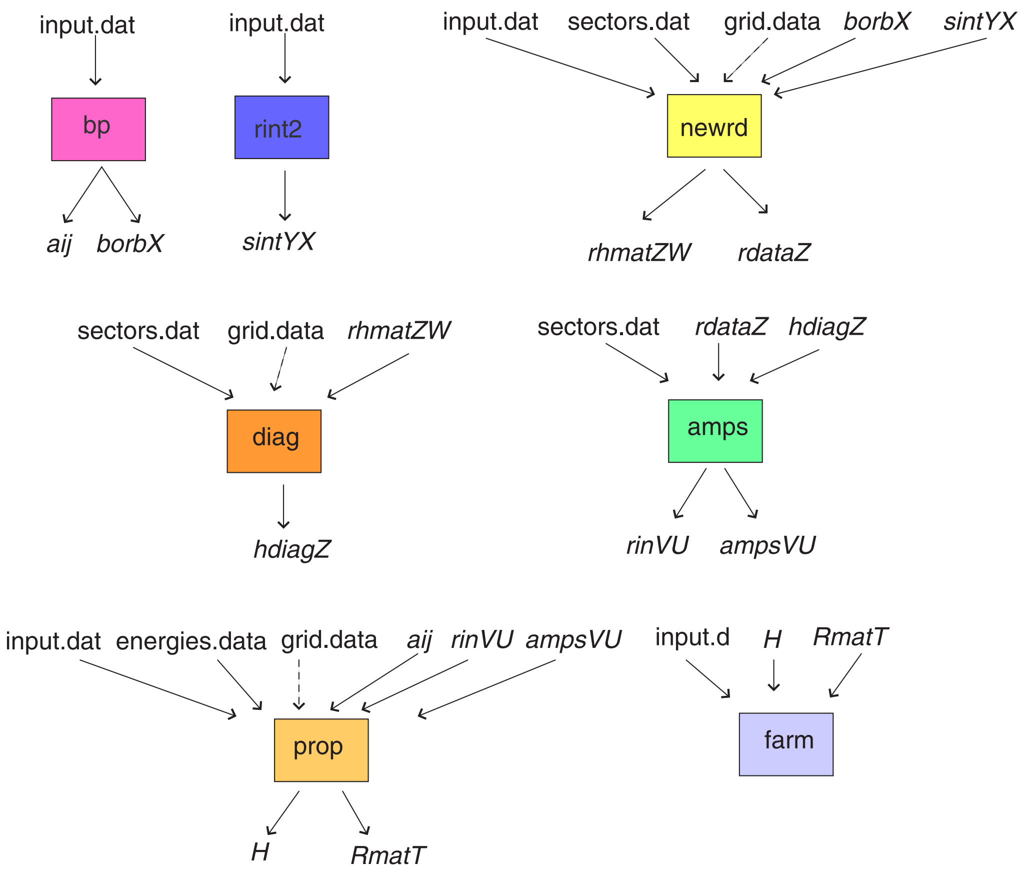


Fig. 8. This figure shows the user supplied input files and program generated output files used to communicate amongst the seven programs identified in Fig. 5. Filenames in italics indicate binary files, other files are text files. $X = 01, 02 \ldots n_{strip}$; $Y =$ '*lt*' or '*gt*'; $Z = 000, 001 \ldots n_{sector} - 1$; $W = 00, 01 \ldots n_{cps} - 1$; $V = 01, 02 \ldots n_{strip}$, $U = 01, 02 \ldots V$, where n_{strip} is the number of subregions across the x-axis (or y-axis), n_{sector} is the total number of subregions and n_{cps} is the number of cores used per subregion; $T = 0000, 0001, \ldots n_{energy} - 1$, where n_{energy} is the number of scattering energies. The input file *grid.data*, indicated by the dashed line, is only used when $n_{cps} > 1$.

```
1:  BP
2:      READS
3:      BNDORBS
4:          MACHIN
5:          SETMESH
6:          PSFINDER1
7:              USOLVE
8:                  DE
9:             NUMNODE
10:            ROOT
11:            ABNORM
12:     BLKORBS
13:         SETMESH
14:         USOLVE [see line 7]
15:         GETORB
16:         ABNORM
17:     AIJCALC
18:         FACTT
19:         SHRIEK
20:         EVALUE
21:             SETMESH
22:         FLAMDA
23:             CG
24:             DRACAH
25:         RAIJ
```

3.1.1. *READS*

Reads the input data supplied through the input file `input.dat` described in Appendix A.

3.1.2. BNDORBS

Solves the eigenvalue problem defined by Eqs. (67)–(68b). Trial eigenvalues are iterated on until the boundary conditions are satisfied: this is controlled by `PSFINDER1`. Eq. (67) is solved using the DE package of Shampine and Gordon [45].

3.1.3. BLKORBS

As required by Eq. (73), Eq. (67) is recomputed, using the eigenvalues found in `BNDORBS`, at the r_2 mesh points across each subregion at boundary $r_1 = r_a$. This subroutine writes the unformatted binary files `borbX` depicted in Fig. 8.

3.1.4. AIJCALC

This subroutine controls the calculation of the long-range potential coefficients, a_{ij}^{λ}, defined by Eqs. (23), (87)–(88). `FLAMDA` computes $f_{\lambda}(l_1, l_2, l_3, l_4 : L)$, defined by Eq. (23), using `CG` and `DRACAH` to compute Clebsch-Gordan and Racah coefficients respectively. `RAIJ` computes I_{ij}^{λ}. In practice, $\lambda_{max} \leqslant 6$ and a_{ij}^{λ} is only used to couple channels involving physical channels, all other a_{ij}^{λ} elements are set to zero. This subroutine writes the unformatted binary file `aij` depicted in Fig. 8.

3.2. Program RINT2

Program 2 (rint2) is primarily concerned with computing information required in off-diagonal subregions. In particular, it computes the *greater than integrals*, $\int_{a_i}^{a_{i+1}} \frac{v_{n_1 l_1}(r) v_{n_3 l_3}(r)}{r^{\lambda+1}}\, dr$, and the *less than integrals*, $\int_{a_i}^{a_{i+1}} v_{n_2 l_2}(r) v_{n_4 l_4}(r) r^{\lambda}\, dr$, for $i = 0, n_{strip} - 1$.

The program also computes information to enable the evaluation of $\mathcal{R}^{c}_{(i,i)}$, defined by Eq. (57a), in off-diagonal subregions. The additional 10 orbitals required for the evaluation of $\mathcal{R}^{c}_{(1,3)}$, $\mathcal{R}^{c}_{(3,1)}$, $\mathcal{R}^{c}_{(2,4)}$ and $\mathcal{R}^{c}_{(4,2)}$ in Eqs. (52a)–(52b) are also computed here.

The integrals and fitting coefficients are written to the binary files `sintgtX` and `sintltX` as shown in Fig. 8. Because of symmetry properties amongst off-diagonal subregions these files contain sufficient information to enable the computation of R_{λ}^{OD}, defined by Eq. (25), and $\mathcal{R}^{c}_{(i,i)}$, defined by Eqs. (57a)–(57b), in all off-diagonal subregions.

```
1: RINT2
2:     READS
3:        INIT
4:        ORBS
5:            MACHIN
6:            PSFINDER
7:                USOLVE
8:                    DE
9:                NUMNODE
10:               ROOT
11:               ABNORM
12:           BUTTLE
13:               USOLVE [see line 7]
14:               LSQ
15:                   MA01A
16:                   PMUL
17:       SKINT
```

The program's call graph is shown above and the key subroutines are described below. Steps (3)–(17) inclusive are repeated for each subregion in the rightmost vertical strip (see Fig. 1).

3.2.1. READS

Reads the input data supplied through the input file `input.dat` described in Appendix A.

3.2.2. INIT

Initialises the integration mesh within a given subregion.

3.2.3. ORBS

Solves the eigenvalue problem, defined by Eqs. (14)–(16b), on a given subregion: as in Program 1 this is controlled by `PSFINDER` and uses the DE package of Shampine and Gordon [45]. For each orbital angular momentum solutions up to principal quantum number $n = n_{max} + 10$ are computed. The additional 10 solutions are needed to compute $\mathcal{R}^{c}_{(1,3)}$, $\mathcal{R}^{c}_{(3,1)}$, $\mathcal{R}^{c}_{(2,4)}$ and $\mathcal{R}^{c}_{(4,2)}$ as described by Eqs. (52a)–(52b).

Subroutine `BUTTLE` is called to compute $\mathcal{R}^{c}_{(i,i)}$, defined by Eq. (57a). This correction is dependent on the channel energy. However, since the correction is slowly varying with respect to channel energy it is approximated by a quadratic using a least squares fit across a range of energies (see Section 7.4.1.5 of [26]),

$$\mathcal{R}^{c}_{(i,i)} = a_0 + a_1 k^2 + a_2 k^4, \quad i \in \{1, 3\}. \tag{89}$$

The least squares fit defined by Eq. (89) is computed in `LSQ`.

3.2.4. SKINT

This subroutine computes the *greater than integral*, $\int_{a_i}^{a_{i+1}} \frac{\nu_{n_1 l_1}(r)\nu_{n_3 l_3}(r)}{r^{\lambda+1}}\,dr$, and the *less than integrals*, $\int_{a_i}^{a_{i+1}} \nu_{n_2 l_2}(r)\nu_{n_4 l_4}(r) r^{\lambda}\,dr$. This information is written to the unformatted binary files `sintgtX` and `sintltX` and supplemented with off-diagonal subregion data in the form of basis function eigenvalues and eigenfunction values on the subregions 4 edges (see Eqs. (14)–(16b)).

3.3. Program NEWRD

The purpose of `NEWRD` is to build $n_{subregion}$ Hamiltonian matrices in given subregions, as described by Section 2.1.1, where $n_{subregion} \geqslant 1$. The subregions to be considered are specified in the file `sectors.dat` which is described in Appendix B. Additionally, when the subregion is a diagonal subregion the A^i coefficients, defined by Eq. (73), are computed, as is information to enable the computation of Buttle corrections as described by Section 2.2.1.

The program's call graph is shown below: steps (2)–(31) inclusive are repeated for each subregion under consideration; steps (3)–(26) inclusive are invoked for diagonal subregions only and steps (28)–(30) are invoked for off-diagonal subregions only. When `USE2MPILAYERS` is set, see Sections 3.3.5 and 3.3.8 below, Step 31 is only invoked by the `master` core within each grid.

```
1: NEWRD
2:     READS
3:     INIT
4:     ORBS
5:         MACHIN
6:         PSFINDER
7:             USOLVE
8:                 DE
9:             NUMNODE
10:            ROOT
11:            ABNORM
12:        BUTTLE
13:            USOLVE [see line 7]
14:            LSQ
15:                MA01A
16:                PMUL
17:    CHECK
18:        ABNORM
19:    FLAMDA
20:        FACTT
21:        SHRIEK
22:        CG
23:        DRACAH
24:    HMATD
25:        HMATDROW
26:            RKINT
27:    PHYSCOEF
28:    READSSKINT
29:    HMATO
30:        HMATOROW
31:    FILWRT
```

3.3.1. READS

Reads the input data supplied through the input file `input.dat` described in Appendix A.

3.3.2. INIT

Initialises the integration mesh within a given diagonal subregion.

3.3.3. ORBS

Solves the eigenvalue problem, defined by Eqs. (14)–(16b), on a given diagonal subregion: as in Program 1 (bp) this is controlled by `PSFINDER` and uses the DE package of Shampine and Gordon [45]. As in Program 1 (bp) information is computed to enable the evaluation of the Buttle corrections as described by Section 2.2.1. Note that on a diagonal subregion edges 1 and 3 are identical to edges 2 and 4, respectively.

3.3.4. FLAMDA

`FLAMDA` computes $f_\lambda(l_1, l_2, l_3, l_4 : L)$, defined by Eq. (23), using `CG` and `DRACAH` to compute Clebsch–Gordan and Racah coefficients respectively.

3.3.5. HMATD

HMATD is used to construct the Hamiltonian matrix on a diagonal subregion, as defined by Eqs. (18)–(21). The size of the Hamiltonian matrix, `ipcount`-1, is determined by the permissible values of $n_1 l_1$ and $n_2 l_2$ in Eq. (8). The permissible values of $n_1 l_1$ and $n_2 l_2$ are restricted by Eqs. (28a)–(28c) and (31)–(32) and stored in arrays ione and itwo, respectively. RKINT is used to compute the two-dimensional radial integrals, R_λ^D, defined by Eq. (24). When USE2MPILAYERS is set the rows of the matrix are distributed, in a load balanced fashion, across the cores within the $n_{row} \times n_{column}$ grid of cores dedicated to the subregion. The rows of the matrix are written to the binary file rhmatZW by each core as described in Fig. 5.

3.3.6. PHYSCOEF

The $A^i_{n_2 l_2 n}$ coefficients, defined by Eq. (73), are computed on diagonal subregions and stored in array acoeff.

3.3.7. READSSKINT

READSRKINT reads the unformatted binary files sintgtX and sintltX corresponding to the off-diagonal subregion under consideration and stores the *greater than integrals* and the *less than integrals* in arrays skint1 and skint2 respectively.

3.3.8. HMATO

HMATO is used to construct the Hamiltonian matrix on an off-diagonal subregion, as defined by Eq. (22). The size of the Hamiltonian matrix, ipcount-1, is determined by the permissible values of $n_1 l_1$ and $n_2 l_2$ in Eq. (8). The permissible values of $n_1 l_1$ and $n_2 l_2$ are restricted by Eqs. (31)–(32) and stored in arrays ione and itwo, respectively. The radial integrals, R_λ^{OD}, defined by Eq. (25), are built from the arrays skint1 and skint2 previously set in READSSKINT. When USE2MPILAYERS is set the rows of the matrix are distributed, in a load balanced fashion, across the cores within the $n_{row} \times n_{column}$ grid of cores dedicated to the subregion. The rows of the matrix are written to the binary file rhmatZW by each core as described in Fig. 8.

3.3.9. FILWRT

Subregion data in the form of basis function eigenvalues and eigenfunction values on the subregions 4 edges (see Eqs. (14)–(16b)), arrays ione and itwo, and Buttle correction data are written to file rdataZ as described in Fig. 8. On diagonal subregions this data is supplemented with array acoeff computed in PHYSCOEF.

3.3.10. The executables

Three executables are built by the makefile: newrd.exe, newrd_1mpi.exe and newrd_2mpi.exe. We shall assume that n_{core} cores are available, where $n_{core} \geqslant 1$.

- newrd.exe: This is a serial code where the $n_{subregion}$ matrices are constructed sequentially.
- newrd_1mpi.exe: The build of this executable is controlled by the macro USEMPI which is set in the makefile. Here each subregion is assigned to a single core with each core processing either $\lfloor \frac{n_{subregion}}{n_{core}} \rfloor$ or $\lceil \frac{n_{subregion}}{n_{core}} \rceil$ subregions.
- newrd_2mpi.exe: The build of this executable is controlled by the macro USE2MPILAYERS which is set in the makefile. In this scenario the n_{core} cores are partitioned into n_{grid} grids, each with $n_{row} \times n_{column}$ cores.[13] Each subregion is assigned to a grid and each grid processes either $\lfloor \frac{n_{subregion}}{n_{grid}} \rfloor$ or $\lceil \frac{n_{subregion}}{n_{grid}} \rceil$ subregions. The topology of the grid is specified in the file grid.data which is described in Appendix C.

3.4. Program DIAG

The purpose of DIAG is to diagonalise $n_{subregion}$ Hamiltonian matrices in specified subregions, thereby generating in each subregion, the eigenvalues, E_k, and the eigenvectors, $\alpha_{n_1 l_1 n_2 l_2, k}$, defined, respectively, by Eqs. (7) and (8). The subregions to be considered are specified in the file sectors.dat. The resulting eigenvalues and eigenvectors are stored in files hdiagZ, where $Z = 000, 001, \ldots n_{sector} - 1$, as described in Fig. 8.

The program uses an object-based programming style and is composed of the following modules which are located in the subdirectory /DIAG: constants.f90, diag.f90, errors.f90, matrix_par.f90, matrix_seq.f90, system_par.f90 and system_seq.f90. Equivalent parallel and serial methods, found respectively in the corresponding _par and _seq modules, have identical method signatures. Accordingly, the system and matrix modules provide an abstract layer that shields the user from needing to know the details of the underlying architecture. The modules are briefly described as follows.

3.4.1. constants.f90

The module defines global constants including file handles.

3.4.2. diag.f90

The driver subroutine.

3.4.3. errors.f90

This module attempts to trap and exit gracefully from fatal and non-fatal errors. These errors take the form of file errors, memory allocation errors and LAPACK and ScaLAPACK routine errors.

3.4.4. matrix_par.f90

This module defines a matrix abstract data type (ADT). It includes types and methods to allow, within a parallel environment: matrix creation; matrix destruction; matrix copying; matrix assignment; matrix distribution; and linear algebra operations including multiplication, addition, inversion and diagonalisation.

[13] $n_{cps} = n_{row} \times n_{column}$ and $n_{core} = n_{grid} \times n_{cps}$.

3.4.5. matrix_seq.f90

This module mirrors `matrix_par.f90` but for a serial environment.

3.4.6. system_par.f90

This module defines a system ADT. It contains private system data including: row and column blocking factors; the core's ID; the number of cores; the number of grids; the grid ID; the grid topology; the grid context. It also provides a collection of methods to set and get this data.

3.4.7. system_seq.f90

This module mirrors `system_par.f90` but for a serial environment.

3.4.8. The executables

Three executables are built by the makefile: diag.exe, diag_1mpi.exe and diag_2mpi.exe. We shall assume that n_{core} cores are available, where $n_{core} \geqslant 1$.

- diag.exe: This is a serial code where the $n_{subregion}$ matrices are constructed sequentially. In this case the serial modules `matrix_seq.f90` and `system_seq.f90` are used in the compilation of the executable. Each subregion Hamiltonian matrix is contained in the single file `rhmatZ00` as described in Fig. 8, where Z indicates the subregion number. Each matrix is diagonalised by a single core using the LAPACK subroutine `xSYEV`.
- diag_1mpi.exe: The build of this executable is controlled by the macro `USEMPI` with the serial modules `matrix_seq.f90` and `system_seq.f90` being used in the compilation of the executable. Each subregion is assigned to a single core with each core processing either $\lfloor\frac{n_{subregion}}{n_{core}}\rfloor$ or $\lceil\frac{n_{subregion}}{n_{core}}\rceil$ subregions. Each matrix is diagonalised by a single core using the LAPACK subroutine `xSYEV`.
- diag_2mpi.exe: In this case the parallel modules `matrix_par.f90` and `system_par.f90` are used in the compilation of the executable. The n_{core} cores are partitioned into n_{grid} grids each with $n_{row} \times n_{column}$ cores.[14] Each subregion is assigned to a grid and each grid processes either $\lfloor\frac{n_{subregion}}{n_{grid}}\rfloor$ or $\lceil\frac{n_{subregion}}{n_{grid}}\rceil$ subregions. The master core within each grid of n_{cps} cores reads the Hamiltonian matrix and block-cyclically distributes it across the grid. The topology of the grid and the blocking factor required by ScaLAPACK is specified in the file `grid.dat`, as described in Appendix C. Diagonalisation is performed using the ScaLAPACK subroutine `PxSYEVD`. On completion the eigenvalues and eigenvectors are copied to the master core which writes them to `hdiagZ`.

3.5. Program AMPS

The primary purpose of `AMPS` is to construct the surface amplitudes defined by Eqs. (40)–(43) across $n_{subregion}$ subregions, where $n_{subregion} \geqslant 1$. The subregions to be considered are specified in the file `sectors.dat` which is described in Appendix B. Additionally, in diagonal subregions the contribution to the C-matrix, defined by Eq. (78), is constructed. The program's call graph is shown below.

```
1: READFILE
if origin subregion
    2: AMPD4
    3: CMATRIX
endif
if axis subregion
    4: AMPO24
    5: AMPO3
endif
if general subregion
    6: AMPO24
    7: AMPO13
endif
if diagonal subregion, other than origin subregion
    8: AMPD24
    9: CMATRIX
endif
```

The key subroutines in `AMPS` are described as follows.

3.5.1. READFILE

This subroutine reads the file `rdataZ` written by `NEWRD` and transfers its contents to `rinVU`.

3.5.2. AMPD4

This subroutine evaluates the surface amplitude, $\omega_{(4)nl_1l_2,k}$, associated with edge 4 (see Fig. 2) in the origin subregion[15] as defined by Eq. (43), with, $a_{nlnl,k}$, given by Eqs. (28a)–(28c). The surface amplitudes are written to the direct access file `ampsVU` as described in Fig. 8.

[14] $n_{cps} = n_{row} \times n_{column}$ and $n_{core} = n_{grid} \times n_{cps}$.
[15] Subregion 0 in Fig. 4.

3.5.3. AMPD24

This subroutine evaluates the surface amplitudes, $\omega_{(2)nl_1l_2,k}$ and $\omega_{(4)nl_1l_2,k}$, associated with edges 2 and 4 (see Fig. 2) in a diagonal subregion[16] as defined by Eqs. (41) and (43), respectively, with, $a_{nlnl,k}$, given by Eqs. (28a)–(28c). The surface amplitudes are written to the direct access file ampsVU as described in Fig. 8.

3.5.4. AMPO3

This subroutine evaluates the surface amplitude, $\omega_{(3)nl_1l_2,k}$, associated with edge 3 (see Fig. 2) in an axis subregion[17] as defined by Eq. (42), with, $a_{nlnl,k}$, given by Eq. (30). The surface amplitudes are written to the direct access file ampsVU as described in Fig. 8.

3.5.5. AMPO13

This subroutine evaluates the surface amplitudes, $\omega_{(1)nl_1l_2,k}$ and $\omega_{(3)nl_1l_2,k}$, associated with edges 1 and 3 (see Fig. 2) in a general subregion[18] as defined by Eqs. (40) and (42), with, $a_{nlnl,k}$, given by Eq. (30). The surface amplitudes are written to the direct access file ampsVU as described in Fig. 8.

3.5.6. AMPO24

This subroutine evaluates the surface amplitudes, $\omega_{(2)nl_1l_2,k}$ and $\omega_{(4)nl_1l_2,k}$, associated with edges 2 and 4 (see Fig. 2) in a general subregion as defined by Eqs. (41) and (43), with, $a_{nlnl,k}$, given by Eq. (30). The surface amplitudes are written to the direct access file ampsVU as described in Fig. 8.

3.5.7. CMATRIX

A^i coefficients constructed in NEWRD are read from the appropriate rdataZ file. Their contribution to the C-matrix, defined by Eq. (78), is constructed. The resulting submatrix of C is appended to rinVU as described in Fig. 8.

3.5.8. The executables

Two executables are built by the makefile: amps.exe and amps_1mpi.exe. We shall assume that n_{core} cores are available, where $n_{core} \geqslant 1$.

- amps.exe: This is a serial code where the surface amplitudes in the $n_{subregion}$ subregions are built sequentially.
- amps_1mpi.exe: The build of this executable is controlled by the USEMPI macro. Here each subregion is assigned to a single core with each core processing either $\lfloor \frac{n_{subregion}}{n_{core}} \rfloor$ or $\lceil \frac{n_{subregion}}{n_{core}} \rceil$ subregions.

3.6. Program PROP

The primary purpose of PROP is to propagate the global R-matrix across the internal-region, as described in Section 2.3, for each of n_{energy} scattering energies. The energies are specified in the file energies.data which is described in Appendix D. As described in Fig. 8, PROP produces a single H file and a collection of RmatT files, one for each scattering energy. The value of T is the index of the corresponding energy in the range defined by energies.data.[19]

The program uses an object-based programming style and is composed of the modules below which are located in the subdirectory /PROP. Equivalent parallel and serial methods, found respectively in the corresponding _par and _seq modules, have identical method signatures. Accordingly, the _par and _seq modules provide an abstract layer that shields the user from needing to know the details of the underlying architecture. The modules are briefly described as follows.

3.6.1. amplitudes.f90

This module is used to read, from ampsVU, the surface amplitudes, $\omega_{(i)nl_1l_2,k}$, described in Eq. (45) as required by the subregion under consideration. A complementary collection of surface amplitudes scaled by $\frac{1}{2(E_k-E)}$ is also computed and stored in array scaled_amps.

3.6.2. axis.f90

This file contains subroutine evaluate_axis_block and is used to compute, $\Re^O$, using Eqs. (64a)–(64d), as the global R-matrix is propagated across an axis subregion. When propagating across the first axis subregion[20] there is no common boundary X and only Eq. (64a) needs to be solved. The file is #included in global_matrix.f90.

3.6.3. block_information.f90

This is an ADT designed to set and access basic data associated with the subregion including its: id, width, total orbital and spin angular momenta, parity, n_{max}, l_{max}.

3.6.4. buttle_corrections.f90

This module computes the two types of Buttle correction. The first, $\mathcal{R}^c_{(1,1)}$, $\mathcal{R}^c_{(2,2)}$, $\mathcal{R}^c_{(3,3)}$ and $\mathcal{R}^c_{(4,4)}$, is defined by Eqs. (57a)–(57b), while the second, $\mathcal{R}^c_{(1,3)}$, $\mathcal{R}^c_{(3,1)}$, $\mathcal{R}^c_{(2,4)}$ and $\mathcal{R}^c_{(4,2)}$, is defined by Eqs. (52a)–(52b).

[16] Subregions $1, 2, 5, 9, \ldots$ in Fig. 4.

[17] Subregions 1, 3, 6, ... in Fig. 4.

[18] Subregions 4, 7, 8, ... in Fig. 4.

[19] Computational note: the local R-matrices in subregion $I_{i,j}$, described by Eq. (45), are scaled by $\frac{1}{a_{j+1}-a_j}$. The physical R-matrix, **R**, described by Eq. (83), is scaled by a further $\frac{1}{n_{sector}}$. This means that the physical R-matrix, **R**, is scaled in total by $\frac{1}{a}$, a being the boundary radius. This is done because the FARM program [43] expects an R-matrix in the form described by Eqs. (25)–(26) of [46].

[20] Subregion 1 in Fig. 4.

3.6.5. *c_matrix.f90*

This module constructs the complete C-matrix defined by Eq. (78).

3.6.6. *channel_info.f90*

This ADT reads the channel data, nl_1l_2, for the subregion under consideration from the file `rinVU`.

3.6.7. *constants.f90*

The module defines global constants for internal use. These include file handles, subregion and edge identifiers.

3.6.8. *diagonal.f90*

This file contains subroutine evaluate_diagonal_block an is used to compute, $\Re^O$, using Eqs. (64a)–(64d) and (65a)–(65d), as the global R-matrix is propagated across a diagonal subregion. The file is #included in `global_matrix.f90`.

3.6.9. *energies.f90*

This module implements an ADT for an energy object which holds the range of scattering energies to be considered and the total energy, E, of the two electron system. This information is derived from data the file `energies.data` which is described in Appendix D.

3.6.10. *errors.f90*

This module attempts to trap and exit gracefully from fatal and non-fatal errors. These errors take the form of file errors, memory allocation errors and LAPACK and ScaLAPACK routine errors.

3.6.11. *files.f90*

This module manages I/O processing to and from external files, `aij`, `rinVU`, `ampsVU`, `H` and `RmatT` as described in Fig. 8.

3.6.12. *global_matrix.f90*

This module handles the storage and processing of the global R-matrix objects, $\Re^I$ and $\Re^O$. This includes the construction of the global R-matrix in the origin subregion and the constructions of the physical R-matrix, **R**, defined by Eq. (83).

3.6.13. *local_rmatrices.f90*

This module handles the construction in each subregion of the local R-matrices, $\mathbf{r}_{II}$, $\mathbf{r}_{IO}$, $\mathbf{r}_{OI}$ and $\mathbf{r}_{OO}$, defined by Eqs. (59a)–(59d).

3.6.14. *matrix_par.f90*

This module defines a matrix abstract data type (ADT). It includes types and methods to allow, within a parallel environment: matrix creation; matrix destruction; matrix copying; matrix assignment; matrix distribution; and linear algebra operations including multiplication, addition, inversion and diagonalization.

3.6.15. *matrix_seq.f90*

This module mirrors `matrix_par.f90` but for a serial environment.

3.6.16. *offdiagonal.f90*

This file contains subroutine evaluate_offdiagonal_block and is used to compute, $\Re^O$, using Eqs. (64a)–(64d), as the global R-matrix is propagated across a general subregion. The file is #included in global_matrix.f90.

3.6.17. *propagator_par.f90*

This is the driver routine for a parallel environment. It controls the propagation described in Section 2.3.

3.6.18. *propagator_seq.f90*

This is the driver routine for a serial environment. It mirrors `propagator_par.f90`.

3.6.19. *states.f90*

This module is primarily concerned with organising data into the format that is required by the FARM program [43]. In particular, the target states and associated channel information are reordered into numerically ascending order before being written to the H-file.

3.6.20. *storage.f90*

This module contains the methods that create and destroy the dynamic storage of objects whose lifetime spans all the subregion computations. There are also destructors for objects whose lifetime spans a single subregion.

3.6.21. *system_par.f90*

This module defines a system ADT. It contains private system data including: row and column blocking factors; the core's ID; the number of cores; the number of grids; the grid ID; the grid topology; the grid context. It also provides a collection of methods to set and get this data.

3.6.22. *system_seq.f90*

This module mirrors `system_par.f90` but for a serial environment.

3.6.23. *timing.f90*

This module provides a class that measures total program and specific subregion execution times within the master process. Timing of the propagation stage is controlled by the `TIMING` macro that is set in the makefile.

3.6.24. *The executables*

Three executables are built by the makefile: prop.exe, prop_1mpi.exe and prop_2mpi.exe. We shall assume that n_{core} cores are available, where $n_{core} \geqslant 1$.

- prop.exe: This is a serial code where the n_{energy} propagations are performed sequentially.
- prop_1mpi.exe: The build of this executable is controlled by the macro `USEMPI`. Here each propagation is assigned to a single core with each core processing either $\lfloor \frac{n_{energy}}{n_{core}} \rfloor$ or $\lceil \frac{n_{energy}}{n_{core}} \rceil$ propagations. Matrix multiplications are performed throughout using BLAS and matrix inversions are performed using LAPACK.
- prop_2mpi.exe: This case the parallel modules *_par.f90 are used in the compilation of the executable. In this scenario the n_{core} cores are partitioned into n_{grid} grids each with $n_{row} \times n_{column}$ cores, i.e. $n_{cps} = n_{row} \times n_{column}$ and $n_{core} = n_{grid} \times n_{cps}$. Each propagation is assigned to a grid and each grid processes either $\lfloor \frac{n_{energy}}{n_{grid}} \rfloor$ or $\lceil \frac{n_{energy}}{n_{grid}} \rceil$ propagations. The topology of the grid is specified in the file `grid.data` which is described in Appendix C. Matrix multiplications are performed throughout using PBLAS and matrix inversions are performed using ScaLAPACK.

3.7. *Program FARM*

To complete the package an asymptotic program, such as FARM [43], is needed to solve the system of equations defined by Eq. (84). The original version of FARM [43] is designed to construct the physical R-matrix, **R**, of Eq. (83), from surface amplitudes contained in the `H`-file. However, in 2DRMP, **R**, has already been constructed for each scattering energy during propagation and each **R** is stored in one of the `RmatT` files described in Fig. 8. Therefore, a modified version of FARM, known as FARM_2DRMP, has been developed *solely* for use with 2DRMP and is published in this issue as a New Version Announcement [42].

FARM_2DRMP contains two codes, `farm.f` and `farm_par.f90`. The former is a serial code while the latter is a parallel F95 code that employs an MPI harness to enable the n_{energy} energies to be computed simultaneously across n_{core} cores, with each core processing either $\lfloor \frac{n_{energy}}{n_{core}} \rfloor$ or $\lceil \frac{n_{energy}}{n_{core}} \rceil$ energies. The input files, `input.d` and `H`, and the output `file farm.out` are as described in [43]. **The energy range specified in `input.d` must match that specified in `energies.data`.** Both codes read **R** directly from `RmatT`.

4. Installation and test run

4.1. *2DRMP installation*

(1) *2DRMP* and *FARM_2DRMP* are distributed as a compressed (gzip) tar files, *AEEA_v1_0.tar.gz* and *ADAZ_v1_1.tar.gz* respectively. Uncompress the tar files and extract their contents, e.g., using the UNIX commands,

```
$ gunzip AEEA_v1_0.tar.gz
$ gunzip ADAZ_v1_1.tar.gz
$ tar xvf AEEA_v1_0.tar
$ tar xvf ADAZ_v1_1.tar
```

(2) Two directories, *AEEA_v1_0/* and *ADAZ_v1_1/*, will be created in the working directory. Within *AEEA_v1_0/* there are the scripts, *makefile.'X'*, *prepare4run*, *tidyup* and the subdirectories, *DOC/*, *SRC/*, *TEST_OUTPUT/*, *inputdata/* and *run.'X'/*, where 'X' corresponds to *hp*, *hpcx* and *hector*, the three systems on which the programs have been tested (see PROGRAM SUMMARY). Within *ADAZ_v1_1/* there is a *FARM/* subdirectory.

(3) Copy the *FARM/* subdirectory to *AEEA_v1_0/SRC/* and change directory to *AEEA_v1_0/*.

```
$ cp -r ADAZ_v1_1/FARM AEEA_v1_0/SRC/
$ cd AEEA_v1_0
```

(4) Construct, or edit, *makefiles.'X'* in the working directory and in *SRC/* and its subdirectories, *DIAG/*, *FARM/*, and *PROP/* appropriate for your system.

(5) Construct, or edit, job submission scripts within *run.'X'/* appropriate for your system.

(6) Execute the script

```
$./prepare4run 'X'
```

This will: create symbolic links from the working directory to data files in */inputdata* and to job submission scripts in *run.'X'/*; create three new subdirectories, *Log/*, *Sector_Data/*, and *propfarm/* used to hold the results when programs are executed; and compile the source code located in */SRC*.

The script *tidyup* can be run at any stage to restore the working directory to its state in (1) above. HTML documentation for *2DRMP*, as described in Section 3, is located in the *DOC/* subdirectory.

4.2. Test runs

The test run selected is for the small but illustrative case of electron scattering from hydrogen, where the hydrogen atom is approximated by its physical $n = 4$ states, $1s, 2s, 3s, 4s, 2p, 3p, 4p, 3d, 4d$, and $4f$, and augmented by numerical pseudo-states with $l \leqslant 4$ and $5 \leqslant n \leqslant 20$, as defined by Eqs. (67)–(68b).

The physical states are known exactly and are required to be read in as analytic functions of the form, $P_{nl}(r) = \sum_{i=1,nt} coef[i] * r^{irad[i]} * e^{-alpha*r}$, with $0 \leqslant l \leqslant \min(lbndmax, nbound - 1)$ and $l + 1 \leqslant n \leqslant nbound$. Here, `lbndmax`, the maximum orbital angular momentum of the target states, is 4, `nbmdmax`, the maximum principal quantum number momentum of the target states, is 20 and `nbound`, the maximum principal quantum number of the physical target-state orbitals, is 4. The R-matrix boundary radius, r_a, chosen to envelope the physical states, is taken as 60 a.u.

The internal region is subdivided as illustrated in Fig. 4, with each subregion being a square with sides of length 15 a.u. This gives four strips and 16 subregions. However, because of the symmetry of the internal region we only need consider the 10 subregions on and below the diagonal. In each subregion we use a maximum of 20 basis functions for each angular momentum (n_{max}). For each total two-electron orbital angular momentum, L, the maximum angular momentum of the one-electron basis functions in each subregion (l_{max}) in an $n = 4$ approximation is taken as $L + 4$. For computational ease we consider the two-electron system to be in the $^1S^e$ state, for which l_{max} is therefore 4. Sixty-four equally spaced scattering energies, between 1.0 and 2.575 inclusive are computed. This is chosen to enable direct comparison with the e-H $^1S^e$ $1s \to 2s$ excitation cross section published in Table 3 of [32].

The data corresponding to this scenario can be found in the file *input.dat*. The format of this file is described in Appendix A.

For the purposes of this these test runs it is assumed that a small parallel system with 16 cores is available. Output from test runs performed on the Queen's University HP cluster [47] can be found in subdirectory */TEST_OUTPUT*.[21]

(1) Block A programs (bp, rint2).
   (a) Invoke the job submission script *run_bp_rint2* to execute the two Block A programs in Fig. 5, *bp* and *rint2*.
   (b) On completion, two output text files, *bp.out* and *rint2.out*, should be found in the subdirectory *Log/*. Check the validity of their output by executing the comparison script,

```
$ cd TEST_OUTPUT
$./compare bp_rint2
$ cd..
```

   (c) Check that the binary data files, *aij*, *borbX*, *sintgtX* and *sintltX*, where $X = 01, 02 \ldots 04$ have been created in subdirectory *Sector_Data/*.

(2) Block B programs (newrd, diag, amps).
   (a) Invoke the job submission script *run_newrd2amps_2mpi*. In this script 16 cores are reserved and 4 cores are devoted to each subregion in *newrd* and *diag*. Accordingly, groups of up to 4 subregions from the 10 are computed simultaneously. The subregions to be considered, and the order of consideration, is defined by the data file *sectors.dat*. The data files *grid.data.newrd* and *grid.data.diag* define the number of cores per subregion and their respective topologies.[22] The program *amps* operates with one core per subregion.
   (b) On completion, the output text files, *newrdX.o*, *diagX.o* and *ampsX.o*, where $X = 000, 001, \ldots 009$, should be located in subdirectory *Log/*. Check the validity of their output by executing the comparison script,

```
$ cd TEST_OUTPUT
$./compare newrd2amps_2mpi
$ cd..
```

   (c) Check that the binary data files, *rdataZ*, *rhmatZW*, *hdiagZ* and *rinVU*, *ampVU*, where $Z = 000, 001, \ldots 009$, $W = 00, 01, \ldots 03$, $V = 01, 02, \ldots 04$, and $U = 01 \ldots V$, have been created in subdirectory *Sector_Data/*.

(3) Block C program (prop).
   (a) Invoke the job submission script *run_prop2mpi*. The propagation program will be executed for the 64 equally spaced scattering energies, between 1.0 and 2.575 inclusive, as defined in the file *energies.data*. In this script 16 cores are reserved and 4 cores are devoted to each propagation. Thus each group of 4 cores will cycle serially through 16 energies, each energy propagation being computed in parallel.
   (b) Check the validity of their output by executing the comparison script,

```
$ cd TEST_OUTPUT
$./compare prop_2mpi
$ cd..
```

   (c) Check that the binary output files, *H* and *Rmat00x*, $x = 0, 63$, are generated and stored in subdirectory *propfarm/*.

(4) Block D program (farm).
   (a) Invoke the job submission the script *run_farm_par*. In this script 16 cores are used to compute the 64 scattering energies, between 1.0 and 2.575 inclusive, as defined in the file, *input.d*.

[21] It is expected that digits in positions 7 and 8, in many output files, will differ across machines and compilers.
[22] 2×2 in this case.

(5) Check of the final results.
   (a) Invoke the script,

```
$./create_text_files 1025
```

   This script has been provided to facilitate checking of the final cross section results. When this script is executed, output from *farm_par* is concatenated into a single file *farm.out*, and all files within *propfarm/* are moved to *propfarm/1025/*. Within this new directory the file *farm.out* is interrogated and the cross section results for transitions, $1s \rightarrow 1s, 2s, 2p, 3s, 3p, 3d, 4s$ and $2p \rightarrow 4d$, deposited in the following text files: *1s1s.txt*, *1s2s.txt*, *1s2p.txt*, *1s3s.txt*, *1s3p.txt*, *1s3d.txt*, *1s4s.txt* and *2p4d.txt*. The energies at which these cross sections are evaluated are copied to *energies.txt*.

(6) Check the validity of the final results by executing the comparison script,

```
$ cd TEST_OUTPUT
$./compare farm_par
$ cd..
```

Similar test runs can be performed for serial computation using the job submission scripts *run_newrd2amps*, *run_prop* and *run_farm* and for parallel computation, where one core is devoted to each subregion in Block C and to a subrange of propagation energies in Block D, using *run_newrd2amps_1mpi* and *run_prop_1mpi* respectively. Corresponding test run output and comparison scripts are included in subdirectory */TEST_OUTPUT*.[23]

5. Concluding remarks

In large scale virtual experiments, involving 200+ sectors, a load imbalance between the construction of the Hamiltonian matrix on diagonal and off-diagonal sectors may be observed. The root of the bottleneck is the large number of two-dimensional radial integrals, the so-called Slater integrals, that are required on each diagonal sector. The problem can be solved using the hand crafted quadrature formula reported in [48]. An enhanced version of 2DRMP including this feature, together with the incorporation of model potentials to extend the target to quasi one-electron atoms and ions, is under development.

Acknowledgements

The authors are grateful to the referee for the very careful reading of the manuscript and for constructive comments.

This research was supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC) under grants EP/F010052/1, EP/C001753/1 and GR/R3118/01. The authors are grateful to EPSRC for the provision of computer time on the UK's national supercomputers HECToR and HPCx, and to Queen's University Belfast (QUB) for computer time on its HP Cluster. TS and VFM acknowledge an ESF/QUB postgraduate studentship. CD acknowledges a research position at QUB, funded by EPSRC, during which part of this work was undertaken.

Appendix A. The input file input.dat

(1) **nstrip**, **rbnd**
   nstrip: the number of strips in the inner region.
   rbnd: the radius, r_a, of the inner region in a.u., rbnd=nstrip*blksize.
   See Fig. 1.
(2) **lbndmax**, **nbndmax**
   lbndmax: the maximum orbital angular momentum of the target states.
   nbndmax: the maximum principal quantum number of the target states.
   See Eqs. (67)–(68b).
(3) **ltot**, **istot**, **npty**, **nz**
   ltot: the total orbital angular momentum (L) of the 2-electron system.
   istot: the total spin (S) of the 2-electron system.
   npty: the total parity of the 2-electron system.
   nz: the nuclear charge of the target atom.
   See Eq. (3).
(4) **npts**
   npts: the number of integration points, which must be odd, in each subregion.
(5) **nmax**, **lmax**
   nmax: the maximum principal quantum number of the basis orbitals.
   lmax: the maximum orbital angular momentum quantum number of the basis orbitals.
   See Eq. (14).
(6) **nbound**
   nbound: the maximum principal quantum number of the physical target-state orbitals, see Section 2.4.
(7) The analytic expansion of the target-state orbitals, i.e. $P_{nl}(r) = \sum_{i=1,nt} coef[i] * r^{irad[i]} * e^{-alpha*r}$, see Section 2.4.

[23] Output from *run_newrd2amps* should be identical to output from *run_newrd2amps_1mpi* and output from *run_prop* should be identical to output from *run_prop_1mpi*.

```
do l=0, min(lbndmax,nbound-1)
    do n=l+1,nbound
        nt
        (irad(l,n,i),i=1,nt)
        (coef(l,n,i),i=1,nt)
        alpha(l,n)
    end do
end do
```

Appendix B. The input file sectors.dat

```
(1) nsectors
(2) do i=0,nsectors-1
        sector_id
    enddo
```

nsectors: The number of subregions to be considered.
sector_id: the subregion identifier.

Appendix C. The input file grid.data

(1) **row_block_factor**
row_block_factor: the block-cyclic row blocking factor, not used in `NEWRD`.
(2) **column_block_factor**
column_block_factor: the block-cyclic column blocking factor, not used in `NEWRD`.
(3) **grid_rows**
grid_rows: the number of rows in the grid of cores.
(4) **grid_columns**
grid_columns: the number of columns in the grid of cores.

Appendix D. The input file energies.data

(1) **initial_energy**
initial_energy: initial scattering energy in Rydbergs.
(2) **final_energy**
final_energy: final scattering energy in Rydbergs.
(3) **energy_increment**
energy_increment: scattering energy increment in Rydbergs.
(4) **target_energy**
target_energy: ground state energy of the target in atomic units.

References

[1] E.P. Wigner, Phys. Rev. 70 (1946) 15.
[2] E.P. Wigner, Phys. Rev. 70 (1946) 606.
[3] P.G. Burke, Computational Physics, Inst. Phys. and Phys. Soc., London, 1970, p. 9.
[4] P.G. Burke, K.A. Berrington (Eds.), Atomic and Molecular Processes and R-Matrix Approach, IOP Publishing, Bristol, 1993.
[5] P.G. Burke, C.J. Noble, V.M. Burke, Adv. At. Mol. Opt. Phys. 54 (2007) 237.
[6] M. Plummer, J.D. Gorfinkiel, J. Tennyson (Eds.), Mathematical and Computational Methods in R-Matrix Theory, CCP2, STFC Daresbury Laboratory, 2007.
[7] D.C.S. Allison, Comput. Phys. Comm. 1 (1969) 16.
[8] P.G. Burke, Comput. Phys. Comm. 1 (1969) 241.
[9] A. Hibbert, Comput. Phys. Comm. 1 (1970) 359.
[10] W.D. Robb, Comput. Phys. Comm. 1 (1970) 457.
[11] W.D. Robb, The calculation of atomic properties, Ph.D. thesis, The Queen's University of Belfast, 1971.
[12] K.A. Berrington, P.G. Burke, J.J. Chang, A.T. Chivers, W.D. Robb, K.T. Taylor, Comput. Phys. Comm. 8 (1974) 149.
[13] K.A. Berrington, P.G. Burke, M. Le Dourneuf, W.D. Robb, K.T. Taylor, V.K. Lan, Comput. Phys. Comm. 14 (1978) 367.
[14] N.S. Scott, K.T. Taylor, Comput. Phys. Comm. 25 (1982) 347.
[15] P.G. Burke, V.M. Burke, N.S. Scott, Comput. Phys. Comm. 69 (1992) 76.
[16] K.A. Berrington, W.B. Eissner, P.H. Norrington, Comput. Phys. Comm. 92 (1995) 290.
[17] K.A. Berrington, P.G. Burke, K. Butler, M.J. Seaton, P.J. Storey, K.T. Taylor, Yu. Yan, J. Phys. B: At. Mol. Phys. 20 (1987) 6379.
[18] M.J. Seaton, J. Phys. B: At. Mol. Phys. 20 (1987) 6363.
[19] D.G. Hummer, K.A. Berrington, W. Eissner, A.K. Pradhan, H.E. Saraph, J.A. Tully, Astron. Astrophys. 279 (1993) 298.
[20] O. Zatsarinny, Comput. Phys. Comm. 174 (2006) 273.
[21] K. Bartschat, E.T. Hudson, M.P. Scott, P.G. Burke, V.M. Burke, J. Phys. B: At. Mol. Opt. Phys. 29 (1996) 115.
[22] K. Bartschat, E.T. Hudson, M.P. Scott, P.G. Burke, V.M. Burke, Phys. Rev. A 54 (1996) R998.
[23] N.R. Badnell, T.W. Gorczyca, J. Phys. B: At. Mol. Opt. Phys. 30 (1997) 2011.
[24] N.R. Badnell, T.W. Gorczyca, J. Phys. B: At. Mol. Opt. Phys. 30 (1997) 3897.
[25] P.G. Burke, C.J. Noble, M.P. Scott, Proc. Roy. Soc. A 410 (1987) 287.
[26] P.G. Burke, M.P. Scott, in: K. Bartschat (Ed.), Computational Atomic Physics: Electron and Positron Collisions with Atoms and Ions, Springer-Verlag, 1996, p. 137 (Ch. 7).
[27] M.P. Scott, T. Stitt, N.S. Scott, P.G. Burke, AIP Conference Proceedings, in: D.H. Madison, M. Schultz (Eds.), Correlations, Polarization, and Ionization in Atomic Systems, vol. 604, AIP Press, 2002, p. 82.
[28] T.T. Schultz, Electron scattering by atomic hydrogen, PhD Thesis, The Queen's University of Belfast, 1990.

[29] M. Le Dourneuf, J.M. Launay, P.G. Burke, J. Phys. B: At. Mol. Phys. 23 (1990) L559.
[30] M.P. Scott, T. Stitt, N.S. Scott, P.G. Burke, J. Phys. B: At. Mol. Opt. Phys. 35 (2002) L323.
[31] B.R. Odgers, $e^- + H$ scattering at Intermediate Energies, Ph.D. Thesis, The Queen's University of Belfast, 1996.
[32] J.W. Heggarty, P.G. Burke, M.P. Scott, N.S. Scott, Comput. Phys. Comm. 114 (1998) 195.
[33] Timothy Stitt, N. Stan Scott, M. Penny Scott, Phil G. Burke, in: J.M.L.M. Palma, et al. (Eds.), Lecture Notes in Computer Science, vol. 256, 2003, p. 354.
[34] T.S. Stitt, Propagation of $\mathfrak{R}$-matrices on a 2D plane, Ph.D. thesis, The Queen's University of Belfast, 2006.
[35] N.S. Scott, L.Gr. Ixaru, C. Denis, F. Jézéquel, J.-M. Chesneaux, M.P. Scott, Trends and Perspectives in Modern Computational Science, Invited Lectures, in: G. Maroulis, T. Simos (Eds.), Lecture Series on Computer and Computational Sciences, vol. 6, 2006, p. 561.
[36] A. Carson, T.H. Harmer, N.S. Scott, V. Faro-Mazo, M.P. Scott, P.G. Burke, in: M. Daydé, et al. (Eds.), Lecture Notes in Computer Science, vol. 3402, 2005, p. 233.
[37] N.S. Scott, V. Faro-Maza, M.P. Scott, T. Harmer, J.-M. Chesneaux, F. Jézéquel, C. Denis, Phys. Particles Nuclei Lett. 5 (2008) 150.
[38] K.M. Dunseath, M. Le Dourneuf, M. Terao-Dunseath, J.M. Launay, Phys. Rev. A 54 (1996) 561.
[39] B.H. Brandsen, C.J. Joachain, Physics of Atoms and Molecules, Longman Publishing Group, 1982.
[40] M.E. Rose, Elementary Theory of Angular Momentum, Wiley, New York, 1957.
[41] P.J.A. Buttle, Phys. Rev. 60 (1967) 719.
[42] V.M. Burke, C.J. Noble, V. Faro-Maza, A. Maniopoulou, N.S. Scott, Comput. Phys. Comm. 180 (12) (2009) 2450–2451, this issue.
[43] V.M. Burke, C.J. Noble, Comput. Phys. Comm. 85 (1995) 471.
[44] Automating Software Documentation with ROBODoc, http://www.xs4all.nl/~rfsber/Robo/, accessed 22 July, 2009.
[45] L.F. Shampine, M.K. Gordon, Computer Solution of Ordinary Differential Equations, W.H. Freeman, San Francisco, 1974.
[46] P.B. Burke, A. Hibbert, W.D. Robb, J. Phys. B: At. Mol. Phys. 4 (1971) 153.
[47] HP Cluster, Itanium II cluster running Red Hat Linux Enterprise AS, Queen's University Belfast, http://www.qub.ac.uk/directorates/InformationServices/Research/HighPerformanceComputing/Services/Hardware/HPResearch/, accessed 22 July, 2009.
[48] L.Gr. Ixaru, N.S. Scott, M.P. Scott, SIAM J. Sci. Comput. 28 (2006) 1252.