PSTGF: Time-independent R-matrix atomic electron-impact code[☆],[☆☆]

L. Fernández-Menchero [a],[*], A.C. Conroy [a], C.P. Ballance [a], N.R. Badnell [b], D.M. Mitnik [c], T.W. Gorczyca [d], M.J. Seaton [e]

[a] Centre of Theoretical Atomic, Molecular and Optical Physics, Queen's University Belfast, University Road, Belfast BT7 1NN, United Kingdom
[b] Department of Physics, University of Strathclyde, 107 Rottenrow East, Glasgow G4 0NG, United Kingdom
[c] Instituto de Astronomía y Física del Espacio, CONICET/UBA, Buenos Aires, Argentina
[d] Department of Physics, Western Michigan University, Kalamazoo, MI 49008-5252, USA
[e] Department of Physics and Astronomy, University College London, Gower St., London WC1E 6BT, United Kingdom

ARTICLE INFO

ABSTRACT

STGF is a community code employed for outer-region R-matrix calculations, describing electron-impact collisional processes. It is widely recognised that the original version of STGF was written by M.J. Seaton in 1983, but through constant refinement over the next decades by worldwide contributors has evolved into its current form that more reflects modern coding practice and current computer architectures. Despite its current wide acceptance, it was never formally published.

Therefore, we present an updated high-performance parallel version of PSTGF, that balances the requirements of small university clusters, yet can exploit the computational power of cutting edge supercomputers. There are many improvements over the original STGF, but most noticeably, the full introduction of MQDT options that provide subsequent integration with ICFT (Intermediate Coupling Frame Transformation) codes, and for either Breit–Pauli/DARC (Dirac Atomic R-matrix Codes), better load balancing, high levels of vectorisation and simplified output. Semantically, the program is full Fortran 90 in conjunction with MPI (Message Passing Interface) though has CUDA Fortran options for the most numerically intensive code sections.

Program summary
Program Title: RMATRX-PSTGF
CPC Library link to program files: http://dx.doi.org/10.17632/3j55fmr86g.1
Licensing provisions: GNU Lesser General Public License v2.1
Programming language: Fortran 90
Nature of problem: The R-matrix outer region code, PSTGF directly calculates various electron-impact driven processes such as excitation and ionisation, or provides K-matrices for input for subsequent ICFT, differential or magnetic sub-level codes. As the problem size increases, there is an associated increase in the input/output, the numerical computation and unbalanced workload, especially for electron-impact energies where the number of open-channels is of a similar size to the number of closed. The code has been significantly modified to address these issues.

PSTGF interfaces the R-matrix inner region with the outer region, with the R-matrix acting as intermediary between the two regions. The outer region expresses an electron moving in the multipole expansion of the target and predominantly employs Coulomb functions, perturbed or otherwise to achieve this. This is a computationally expensive task, as the R-matrix must be formed for every energy point of every partial wave.
Solution method: An approach that permutes both the partial wave and energy of the incident electron has been implemented. In this version, each processor does not calculate the same incident energy point for each partial wave, but rather distributes all energy points across all processors. This achieves better load-balancing of the work between cores and avoids the case where an overloaded single processor has to always calculate in the energy range where there are approximately the same number of half-open or half-closed channels, which is numerically intensive.

---

[1] M.J. Seaton, Coulomb functions for attractive and repulsive potentials and for positive and negative energies, Comp. Phys. Comm. 146 (2) (2002) 225–249. doi:https://doi.org/10.1016/S0010-4655(02)00275-8.
[2] M.J. Seaton, Quantum defect theory, Rep. Prog. Phys. 46 (2) (1983) 167. doi:https://doi.org/10.1088/0034-4885/46/2/002.

1. Introduction

Time-independent R-matrix theory [1] is a powerful formalism that may be used to calculate first order collisional processes, order α, where usually there is one single active electron in the system: electron-impact excitation, ionisation, photoionisation, dielectronic recombination, radiative recombination for atoms and their associated ion stages. In its simplest form, R-matrix theory splits the collisional problem into two distinct regions: an inner and an outer region. The inner region is defined by the radial extent of the most diffuse orbital from the nucleus. This is sufficient to encompass the charge cloud of the target, though mathematically this R-matrix radius may be extended further, though at a computational cost. The inner region represents target atom/ion plus incident/outgoing electron, and is treated as $N + 1$ electron many body problem, with all the interactions taken in account between indistinguishable particles, including exchange and correlation. We calculate a complete set of eigenfunctions representing both the bound and continuum spectrum. One of the strengths of the R-matrix approach is that as we achieve a complete description of the system in the inner region that is independent of the incident electron energy, therefore only requiring one diagonalisation of $N + 1$ Hamiltonian, for a range of electron-impact energies. The set of codes used for the calculations in the inner region was published in [2–5] and although they form a foundation for this work, they are not the subject of present paper.

In the outer region, the formalism is quite different, the problem is treated as a single electron moving in a multipole expansion of the target. In PSTGF this single electron, which is moving in the potential created by the remaining electrons, is considered to be at a distance that correlation effects would be minimal, and therefore are not included. Comparatively, in this region the physics simplifies, but its range may extend to large distances due to the long-range effect of Coulomb-like potentials. To extract collision strengths, we must be at distances from the nucleus where the wave function has returned to well-known asymptotic forms, a distance considerably larger than the R-matrix inner-region radius.

The key of R-matrix method is the connection of the wave function at the boundary between the inner and outer regions, as this interface determines the phase shifts, consequently the **S** matrices, and finally the collision strengths. Greater details about R-matrix method are given in [1].

PSTGF is the most common atomic outer-region program used to perform the calculations in the outer region for electron-impact excitation processes. As well as directly calculating electron-impact excitation collision strengths, it is the progenitor code for subsequent ICFT, LS, jK and jj differential codes, as well as magnetic sub-level work. It has the capabilities to be interfaced with the molecular suite of R-matrix codes. In general,

PSTGF can be used for any process that can be described by the time-independent and non-damped R-matrix formalism, the most common process included in such theory is the electron-impact excitation and de-excitation of atoms and ions; other important processes are electron-impact ionisation, single or multiple, if the target structure includes such ionising states, electron-impact excitation of molecules, or any one-electron process described by time independent R-matrix theory.

PSTGF is a community code, employed by the majority of R-matrix groups around the world, many of whom have made contributions and improvements over the decades. In present work, we build upon this to provide new sustainable version, that will address future problems that inevitably will be larger in scale.

Below, are some notable additions to the code have been carried out over the years, that due to the original code not being formally published may have been overlooked. These include the extension to neutral atoms [6], the ICFT (Intermediate Frame Transformation) by Griffin and co-workers [7], and the first systematic parallelisation of the serial code by T.W. Gorczyca [8] and D.M. Mitnik (2002, unpublished), using the MPI protocol. This version tried to balance the work-load by distributing the impact-electron energy distribution across the whole energy range for each processor, and remains effective when the number of processors is significantly less than the number of energy points. However, in the intervening years, the number of cores available to researchers has increased to the point where each processor only carries out 1–3 energy points per partial wave. This spurs some of the work presented later in this paper.

Subsequently, this parallel PSTGF version was improved in efficiency. In this new version, PSTGF was restructured in terms of its memory usage. At every point in the calculation, memory is assessed and if not required deallocated, before the next section reducing the memory footprint of the code by half. Additionally, the _H.DAT_ input file was split into individual partial waves _H.DATXXX_ files, that ensured that thousands of processors no longer needed to read a single file, competing against each other, but could concurrently read 40–100 files in smaller groups of processors. Where possible, the LAPACK routines for matrix–matrix and matrix–vector multiplication where employed, especially if an optimised-vendor supplied library was available. Specific routines that considered the multipole perturbation of the non-perturbed results were heavily loop unrolled and refactored to ensure the greatest degree of vectorisation. However, even these improvements need further consideration if we are to progress routinely to systems involving Hamiltonian matrices in excess of 100 000 by 100 000 and involving over 10 000 channels. In this paper, we describe how to address these problems, ranging from removing the last vestiges of Fortran 77 legacy code (COMMON blocks, hard-dimension arrays), to introducing GPU enabled sections for future hardware compatibility.

2. Glossary of terms

- Target state: eigenfunction of the Hamiltonian of the target, N-electron system. It can be labeled with quantum numbers appropriate to the coupling scheme employed. Usually the electronic orbitals are calculated by other specialised packages, for example AUTOSTRUCTURE [9], MCHF [10], GRASP [11,12], or CIV3 [13]. R-matrix inner region codes use these orbitals to calculate the target energies and eigenfunctions.

- Partial wave $J\pi$ or $LS\pi$: symmetry of the initial system target state plus incoming electron, the quantum numbers are conserved. Each Hamiltonian representing a partial wave is calculated independently of every other partial wave. The total cross section is then the sum of the partial cross sections for all possible values of J/L, from zero to infinity, all possible couplings of S (LS coupling case), and π, even and odd.

  For practical reasons, we have used for present work the notation of the relativistic coupling $J\pi$, but the whole procedure is equivalent for the non-relativistic case, and is achieved by just substituting the indexes $J\pi$ for $LS\pi$.

- Channel: In the inner region, it is an eigenfunction of the Hamiltonian for the $N + 1$-electron system. They are orthogonal to each other. The channels conserve the quantum numbers of the total angular momentum J and parity π, in the case we are working in level-resolved, relativistic $J\pi$ coupling; or orbital angular momentum L, total spin S and parity π in the case we are working in term-resolved non relativistic LS coupling. Hence, channels are associated to the partial waves. Channels can be:

  - Open: a channel is classified as open when the energy of the incident electron exceeds the energy between an initial term/level and a final term/level. In this case, a transition may be produced, the final population of the channel will be larger than zero. It has an oscillatory asymptotic form.
  - Closed: A channel is classified as closed when the incident electron energy is less than the energy between and initial term/level and a final term/level. In this case, the transition is energetically impossible. It has an exponentially decaying asymptotic form.

- Phase shift $\delta_{J\pi}$: phase shift of the wave function in the asymptotic region with respect to the case of a pure Coulomb potential, in the case of ion target; or constant potential, in the case of neutral target. Transition matrices, and in consequence cross sections, may be determined in terms of the phase shifts.

- Transitions matrix $T_{if}^{J\pi}$, from initial state i to final state f: transition amplitudes. Its module square $|T_{if}|^2$ is the transition probability from the initial target state i to the final one f. The calculation of the transition matrix is particular and independent for each partial wave. $|T_{if}|^2 = \sum_{J\pi} |T_{if}^{J\pi}|^2$

- Collision strength $\Omega(i - f)$, from initial state i to final state f. Dimensionless version of the cross sections, see [1] for details. $\Omega(i - f) = \sum_{J\pi} \Omega^{J\pi}(i - f)$

- R-matrix $\mathbf{R}^{J\pi}$: A matrix ($n_{channels} \times n_{channels}$) which connects the wave functions of the channels between the inner and the outer region. This matrix is the key to calculate the phase shifts, and with them the transition matrices, collision strengths and cross sections. See [1] for theory details.

3. Overview

PSTGF v1.1 beta 2019 is an upgrade of the previous working version v0.87.[1] The program is implemented in Fortran 90 language, parallelised with the message passing interface (MPI) protocol, and with optional CUDA features for further optimisations if the computer architecture allows. PSTGF reads as input the target state eigen-energies, the channels associated with each target state for each partial wave, and the surface amplitudes (acquired from the matrix diagonalisation) for each partial wave. All of this information is encapsulated in the *H.DATXXX* file. PSTGF enforces the continuity of the radial wave function and its first derivative between the inner and outer regions via the $\mathbf{R}$ matrix. The main goal is the solution of the Schrödinger equation for one electron under the potential created by the multipole expansion potential of the others, as a single particle model, neglecting the electron exchange. This may be expressed as

$$\left[-\frac{1}{2} \frac{d^2}{dr^2} + \frac{l(l+1)}{r^2} + \frac{2z}{r} + \epsilon_n \right] F_n^{J\pi} = \sum_{m=1}^{N} U_{nm}^{J\pi} F_m^{J\pi}, \qquad (1)$$

with n from 1 to N, number of channels in the partial wave $J\pi$, r is the radial coordinate, l is the orbital angular momentum, which depends on the partial wave $J\pi$, $V = \frac{2z}{r}$ the potential in the outer region; $z = Z - N_e$ is the effective charge. $\epsilon_n = E - e_n$ is the channel reduced energy, being E the impact energy of the projectile, and e_n the excitation energy of the individual target level, hence $\epsilon_n > 0$ determines an open channel, while $\epsilon_n < 0$ a closed one. F_n is the radial wave function, and U_{nm} is the long-range multipole expansion.

Once Eq. (1) has been solved, the radial wave function F_n is calculated from the asymptotic region to the interface between the inner and outer regions r_0, then we have to match the solution with that from the inner region to fulfill its continuity. This is achieved by the $\mathbf{R}$ matrix, which performs the unitary transformation among the channels to fulfill the continuity of the radial function and its first derivative

$$F_n(r_0) = \sum_{m=1}^{N} R_{nm} \left(r_0 \left. \frac{dF_m}{dr} \right|_{r=r_0} - b F_m \right), \qquad (2)$$

b is defined by the boundary conditions in the inner region, as the value of the logarithmic derivative of the radial wave function in the interface between inner and outer regions. This is usually chosen to be zero.

The surface amplitudes w_{nk} are defined as follows:

$$w_{nk} = \sum_{j=1}^{n_c} c_{njk} u_{nj}(r_0), \qquad (3)$$

where $u_{nj}(r_0)$ are the reduced surface amplitudes in the inner region; and c_{njk} coefficients obtained from the $N + 1$ Hamiltonian diagonalisation. The summation extends to the size of the continuum basis for each orbital n_c, n extends over the number of channels, while k the second index is over the Hamiltonian matrix size. The $\mathbf{R}$ matrix itself can be defined as follows:

$$R_{nm} = \frac{1}{2r_0} \sum_{k=1}^{M} \frac{w_{nk} w_{mk}}{E_k - E}, \qquad (4)$$

where the E_k are the R-matrix poles or eigenvalues of the $N + 1$ system. This expression (4) was traditionally the part of the calculation which took the most of the time, but now it has been highly optimised using the CUDA programming techniques

---

for GPU (Graphical Processing Units). For further mathematical details we refer to [1,14,15].

Note in Eq. (1) that the reduced energy ϵ_n, greater or lower than zero, will determine the character of the channel, open or closed, so a different asymptotic behavior, see Section 2. The relative number of open and closed channels for a value of the impact energy will be relevant in terms of the computation, we designate as n_o the number of open channels and n_c the number of closed channels. Eq. (1) varies slightly if written in a different coupling scheme i.e., Dirac R-matrix calculations replace l with κ. Eq. (1) has to be solved for all the values of the impact energy E and the complete set of partial waves $J\pi$, usually several thousands of times. For ions, the collision strengths versus the impact energy present narrow Rydberg resonance structures and as a consequence the grid of the impact energies has to be quite fine to delineate them.

Use of PSTGF assumes that the eigen-energies and eigen-functions in the inner region have been previously calculated, so we know all the R-matrix poles E_k and coefficients w_{nk} for all the partial waves. This can be done with several methods and software packages; some examples are RMATRX [3], DARC [4,5], or BSR [16]. All of the information about the channels in the inner region (E_n and w_{nk}) is stored in a set of generic binary files *H.DATXXX*. These may be concatenated into a single *H.DAT* file, used by previous versions of the code, though for good optimisation we would advise against this. Ideally, the inner region has diagonalised every Hamiltonian concurrently and the Hamiltonians are already in this *H.DATXXX* form. PSTGF requires as input the values of the wave functions of all the channels at the boundary of the inner region r_0 for all the partial waves. With these initial conditions, PSTGF expands the wave function in terms of Eq. (1) from r_0 up to a certain asymptotic limit r_1, in which the wave function can be replaced by its analytic form, of a Coulomb function in the case of a charged target, or a spherical Bessel function in the case of the neutral target. From this point onwards, the wave function $F_n^{J\pi}$ just follows its analytical asymptotic solution. In the interface between the inner and outer regions r_0, we have to impose the continuity of the wave function and its first derivative.

The main calculation is distributed in two nested iterative loops. The outer loop runs over partial waves $LS\pi$ or $J\pi$, PSTGF has input options (*dstgf*) to calculate all the partial waves stored in *H.DATXXX* files or to restrict the calculation to a subset of them. To obtain the final collision strengths Ω, PSTGF has to sum up all the contributing partial ones $\Omega_{J\pi}$, obtained for each partial wave. The inner loop concerns the impact energies E, this loop runs over a set of *NE* discrete values of the scaled energy E_k. This grid should be fine enough to delineate fine Rydberg resonance structure, and therefore requires a minimum several thousand energy points. The inner loop, in energy, is the one which is parallelised, the energy array $\{E_k\}$ is split among the processors *nproc*, so each processor is assigned a set of *nproce = NE/nproc* energies to calculate. For an optimum performance, the number of energies should be an even divisor of *nproc*, if this is not the case, then PSTGF will add additional points to enforce this.

The first step in the calculation is to determine which channels are open and which ones are closed, so their asymptotic behavior is set as boundary condition to Eq. (1). Then, PSTGF performs a Numerov method to propagate the coupled radial wave functions $F_{nm}^{J\pi}$ from the boundary of the inner region r_0 and the asymptotic limit r_1. An overview of the operating mode of PSTGF is as follows:

1. Read standard input: energy grid E, partial waves $J\pi$ to be processed, other calculation parameters.
2. Read information about the target *H.DAT*, $J\pi$ independent.
3. Start loop in $J\pi$.
4. Read information about the partial wave $J\pi$ *H.DAT*.
5. Start loop in E. Parallel, split all the E values in all the processors.
6. Calculate partial $\Omega_{J\pi}$ for each energy.
7. End loop in E.
8. End loop in $J\pi$.
9. Add up the partial $\Omega_{J\pi}$ for all partial waves, add the top-up up to $J \to \infty$, and get the total $\Omega(E)$.
10. Write output to file *OMEGA*.

4. Computational details

As computer hardware capabilities have improved, more complicated systems have been undertaken, resulting in increasing number of symmetries and more channels being calculated by PSTGF, which revealed some implementation issues. The first problem detected was the large time differences based upon different incident electron energies. The key understanding in this issue concerns the number of open and closed channels n_o, n_c. It was detected that when both were approximately equal $n_o \sim n_c$, the computation time increases dramatically in comparison to the cases where either only a few channels were open, or all channels were open. Therefore, in the input array of impact energies E_k, there is a distinction between what we call "fast energies" and "slow energies". Usually all the partial waves include channel energies up to a certain threshold E_{max}, common for all partial waves. Up to now, for a fixed impact energy, all the partial waves will have a similar number of open and closed channels. In other words, if an energy is 'fast' or 'slow' for an individual partial wave, most probably, it will have the same character for all of them. However, computationally the bottleneck lies in the energies around those with half the channels being closed and the other half open. In previous v0.87 version, the energy array to be calculated by each processor was independent of the partial wave, and all the processors worked on the same energies for each and every partial wave. When the number of processors increases that gave rise to some of the processors which only had a small number of slow energies, or even none, while other processors had to work considerably harder for several slow energies for all the partial waves. Hence, the time distribution among the processors was very different and the fastest ones remained idle for large parts of the calculation, while waiting for the few slow ones. This problem is maximised when the number of processors increases, so there are less energies per processor to proceed. In new version v1.1, we fix this issue and improve the time balancing by having the energy grid different for each partial wave, so the amount of fast energies and slow energies calculated by each processor is evenly distributed. Fig. 1 shows a diagram of this energy distribution to the processors, in the left picture (v0.87) each processor works only an energy array for all partial waves, while in the right picture (v1.1) each processor works all the energies.

In v0.87 all the processors worked the same energies for all the partial waves, and the output was split into several *OMEGAXXXX* files, *XXXX* being the index of the processor, from 0000 to 9999. Each processor was assigned a unique output file, containing unique calculated energies which via a post-processing code, sorted and collated each file into a single universal *OMEGA* file of ascending incident energy values. In v1.1 all processors calculate all energies, but never consecutive energies for the same partial wave. This mitigates the issue of the 'slow' and 'fast' energies, as no individual processor is assigned the half-closed/half-open channel energy for all partial waves, this now becomes shared among all processors. As consequence, each *OMEGAXXXX* will not contain a converged collision strength at each energy point, in fact each *OMEGAXXXX* file now containing every energy only has
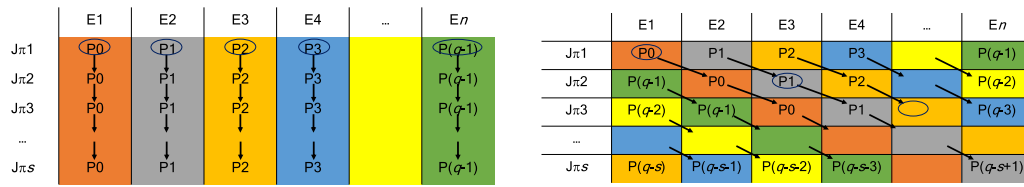
Fig. 1. Diagram of energies and symmetries worked by each processor. Left: v0.87; right v1.1. Pi represents the processor, it is also represented by a particular color; Ei represents the energy array to be processed by processor Pi in the partial wave $J\pi$. The oval represents the partial wave were the processor Pi starts calculating. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

meaning when summed over the respective energy points in all files. Consequently, the *OMEGA* output file cannot be split, and therefore the partial collision strengths are added up inside PSTGF using the MPI routine MPI_REDUCE, and a single *OMEGA* output file is produced.

A second lack of efficiency of previous versions to v0.87 was the reading of the input file *H.DAT*. If the number of partial waves and channels increases, the size of the *H.DAT* file can reach several gigabytes. In previous versions of PSTGF, this single file had to be read by all the processors each time they started the calculation for a new partial wave. It was usual when the program was started, that all the processors attempted to read the same large file at the same time, inevitably leading to some processors blocking others, wasting computing time. Version v0.87 had already implemented the possibility to work beyond a single large *H.DAT* file, by splitting the monolithic single *H.DAT* into several smaller *H.DATXXX* files, being *XXX* the label of the partial wave, from 000 to 999. Then for each partial wave, the processor must read just the individual *H.DATXXX* which contains the information about it, and not the whole *H.DAT* file, with several records of irrelevant information. Nevertheless, it was still a blocking issue when all the processors attempted to read the same file for the same partial wave, especially at the starting of the calculation. As mentioned above we solved this problem by changing the order that the processors work for the partial waves. In v1.1 all the processors carry-out all the same partial waves, but with the difference being, not in the same order. This implementation avoids all the processors reading from the same file at the same time, so these reading waiting queues are minimised. We avoid using MPI communications inside the loop in the symmetry, because it would set barriers, and that would make inefficient the work load balancing among the processors.

Finally, other programming semantic improvements have been performed. All COMMON blocks have been removed and replaced by MODULE, so the programming style is clearly FORTRAN90. All the statically dimensioned arrays and matrices have been removed, previously enforced through a parameter file (*PARAM*). Now, all arrays/matrices are designated allocatable and their scope is dynamically assigned based upon the values read from the *H.DATXXX* files. With this change, PSTGF does not suffer from unexpected segmentation faults, due to a badly user defined variable in the *PARAM* file. In addition, the memory usage is minimised to the essential required. The calls to auxiliary routines from libraries LAPACK and BLAS have been strategically modified in order to take advantage of their optimisation. Another change carried out in the subroutines that contain the most load of work for PSTGF, is to split the memory in an strategic efficient way to work at cache level, with faster access.

In principle, any outer-region electron-impact excitation calculation that can be carried out with present version of PSTGF would also be possible with previous parallel versions, or even the serial version STGF, considering the calculation time can become huge. The new version of the code will work without doing any changes to the input files. Nevertheless, some minimum and optional changes can be done, and they will improve the efficiency of new version:
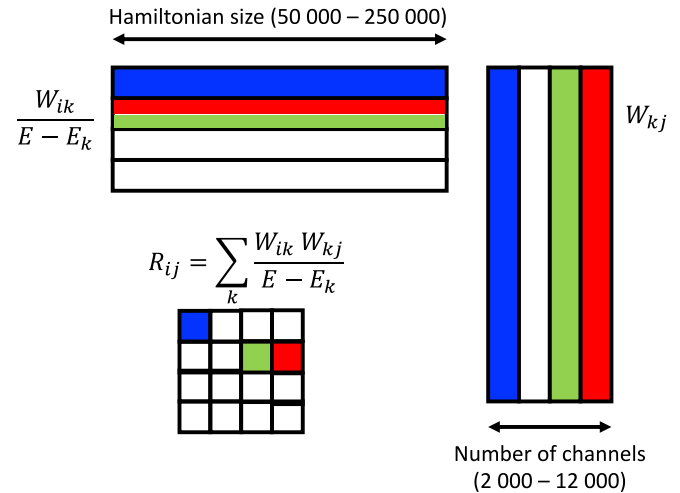

Fig. 2. Half of the symmetric R-matrix is calculated by 10 matrix multiplications, and then symmetrised for the full result. We optimise the memory usage of the GPU, by allowing several MPI tasks to access the GPU simultaneously. In the figure, the blue, red and green blocks are representative of a particular task of matrix multiplication sent to the GPU, three in present case. In general, each task calculates in a different energy and partial wave, and hence the submatrix of a different R-matrix. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- Remove the old parameter dimension information file *PARAM*. The code no longer uses it, it is absolutely redundant, as all the dimensions are now dynamically allocatable.
- The former parameter to set the maximum memory storage dimension without opening any scratch file, MZMEG, is now an input variable. Set its value in the namelist STGF of the standard input if you need to use scratch files instead of in-memory storage (default value is no scratch file to be opened). We strongly recommend against that unless it is really necessary, the hard disk can get exhausted if the number of processors is large.
- If the inner-region code provides one unique *H.DAT* file, split it into several files *H.DATXXX*, each one containing one only partial wave. We recommend to use the utility tool HSPLIT.
- Keep the partial-wave list file *sizeH.dat* or *sizeBP.dat*, and format it properly, so the order of the partial waves in this file is the same as the order in the *H.DATXXX* files.

5. CUDA optimisations for PSTGF

As the processor speed of individual CPUs has remained largely stagnant for the last decade, one option to maintain greater scalability of existing codes is to interface with the power of GPUs (graphical processing units). CUDA Fortran combined with Nvidia GPUs is one of the simplest ways to seamlessly harness the power of MPI and the GPUs. GPU usage is ideally suited to dense matrix multiplications, and the initialisation of the R-matrix falls into this category. As illustrated in Fig. 2, the R-matrix

can be calculated in a single matrix multiplication, but this would exhaust the total memory of the GPU by a single processor. A more optimal use involves separating the R-matrix formation into ten smaller matrix multiplications, essentially pipelining them, one after another. The collective result from these 10 matrix multiplications, can be then symmetrised to provide the full result. This greatly reduces the consumption of memory on the GPU and allows more MPI tasks to concurrently access it. The different colored blocks, and the respective positioning in the resulting R-matrix are given in Fig. 2, and corresponds to 3 different R-matrix matrices, at 3 different energies, being carried out concurrently. Actual matrix multiples are further optimised by padding the larger matrices with zeros to ensure that the matrices fall on optimal, divisible by 32 or 64 boundaries.

Traditionally, for medium to small cases the most time-consuming aspect of the code was the formation of the R-matrix itself (subroutine RINIT). Test cases for neutral Fe, show that for 7261 channels and a Hamiltonian size of 106 661 takes 14–18 secs per R-matrix formation and achieves a factor of 80–100 speed-up over existing code.

6. Code structure

6.1. Input files

6.1.1. Standard input

File *dstgf*, mandatory, ASCII.

Structured in NAMELIST blocks, and optionally additional parameters.

NAMELIST STGF **General calculation parameters**.

IPRINT   Integer from -2 to 3. Default -2.
Moderates the level of standard output (*routf*). Principally used for debugging
   -2 minimum essential output information.
   3   maximum available output information. The size of the output file can become very large.

IPERT   Integer from 0 to 4. Default 0.
To include the long-range multipole potentials. See QDT-variable section for more details.
   0   Omit the long-range multipole potentials.
   1, 2   Omit long-range multipole potentials for closed channels when their asymptotic tail extends outside the asymptotic limit r_1.
   3, 4   Include all the long-range multipole potentials, consider the contribution from r_1 to infinity.
   1, 3   Perturbation for the T matrix.
   2, 4   Perturbation for the K matrix.

PERT   Character(3). Default " ".
Activates the perturbation, variable IPERT automatically. Useful option if QDT is not activated. This variable should not be included in the namelist if IPERT is set manually.
   "YES"/"yes"   Activate perturbation, set IPERT= 4.
   Otherwise   Do not activate perturbation, set IPERT= 0.

AC   Float positive. Default 1.E-5.
Accuracy/tolerance required for some numeric subroutines.

IMESH   Integer, negative or from 1 to 3. Mandatory, no default value.
Defines the energy mesh. See namelist IMESH1, IMESH2, IMESH3 for details.
   1   For fixed linear grid of incremental energies.
   2   For fixed non-linear grid based upon effective quantum number (not recommended).
   3   Read directly the energy mesh from standard input. See additional variables section.
   $-(2S+1)$ To chose appropriate mesh for a case of spin S (not recommended).

IOPT1   Integer, 1, 2, -1, -2, 10, 11. Default 1.
Defines the partial waves $SL\pi$ or $J\pi$ to be calculated.
   1   Calculate all the symmetries stored in the *H.DATXXX* files.
   2   Specify directly in the standard input the symmetries to be calculated. See below format in additional variables section.
   -1 /-2 As options =1, 2, but in this case treat target levels as degenerate, specify the degenerate levels after namelist as additional variables, see such section and NASTD parameter.
   10/11   For JAJOM, obsolete and not implemented in parallel code.

MINLT   Integer. Default -1.
MAXLT   Integer. Default 1000.
Case IOPT1= 1, not used in case IOPT1= 2. Operate only the symmetries which fulfill $L, 2J \geq$ MINLT and $L, 2J \leq$ MAXLT. The default values indicate that all the partial waves included in the *H.DATXXX* files will be calculated.

IRDEC   Integer, 0, 1, 2. Default 0.
Calculate the radiative decays between all the target terms or levels connected by an electric dipole transition.
   0   No radiative decays are calculated.
   1   No QDT calculation, Bell and Seaton radiative decays.
   2   If MQDT is activated, Hickman–Robicheaux radiative decays.

LRGLAM   Integer. Default -1.
Activates the high-L top-up. If top-up is activated then the partial waves must be calculated in ascending order by all the processors, so the efficiency is reduced. It is recommended to use the top-up option in a single calculation including only the two last partial waves.
   Negative or 0   No top-up.
   Positive   Perform top-up from the highest calculated partial wave up to infinity.

LCBE   Integer. Default =LRGLAM.
        Advanced control parameter for the dipole top-up.
        Not recommended to modify its default value, only
        for advanced users.

ITOP   Integer. Default -1.
        Controls the non-dipole top-up over degenerate
        states. Not necessary if IOPT1 is positive. Not
        recommended to modify.
        -1   Interpolate between degenerate and
                non-degenerate limits when energy ratio
                exceeds 2L.
        Other   Interpolate between degenerate and
                non-degenerate limits when energy ratio
                exceeds the J ratio.

ELAS   Character(3). Default " ".
        Controls if the elastic collision strengths should
        be written to output OMEGA file.
        "YES"/"yes"   Write elastic transitions to OMEGA file
                       in any case.
        "NO "/"no "   Do not write elastic transitions to
                       OMEGA file in any case.
        Otherwise   Write the elastic transitions to OMEGA
                       file if the target is neutral, or do not if
                       it is an ion.

IPRKM   Integer from 0 to 4. Default 0.
        Print **K** matrices in optional output file
        KMAT.DAT. For ICFT method. Not recommended
        to use this option, ICFT code is not prepared for
        present implementation of PSTGF. To work in
        ICFT formalism, use previous v0.87 version.
        0   Do not write **K** matrices.
        1   Each processor writes a binary, sequential file
                KMAT.DATXXXX with the physical **K** matrix
                elements, being XXXX the processor number.
        2   Write unphysical **K** matrix to KMATLS file. Not
                implemented in parallel code.
        3   For input of post-processing differential cross
                sections. This option has been removed from
                code and no longer available.
        4   Each processor writes unphysical **K** matrices to
                KMTLS.YYY.XXXX files, if QDT is not activated, or
                S matrices to SMTLS.YYY.XXXX files, if QDT is
                activated. These files are input for ICFT code.
                YYY is the number of symmetry, and XXXX the
                number of processor.

IDIP   Integer 0 or 1. Default 0.
        Write the target dipole electric line strengths
        0   Do not write the line strengths
        1   Write the dipole line strengths S to output
                ASCII file STRENGTH.DAT.

NOMWRT Integer. Default, see below.
        Output management to file OMEGA
        0   Collision strengths are not written to any file.
        > 0   Collision strengths Ω for first NOMWRT
                transitions are written to output file OMEGA as
                upper triangle row-wise.

        < 0   Collision strengths Ω for first -NOMWRT
                transitions are written to output file OMEGA as
                upper triangle column-wise.
        The default value indicates that all the
        transitions among the NAST target levels will be
        stored in the OMEGA file row-wise,
        NOMWRT=((NAST*(NAST-1))/2 if elastic transitions
        are not included, or NOMWRT=((NAST*(NAST+1))/2
        if they are.

IBIGE   Integer. Default 0.
        Print the infinite energy limit collision strengths
        for the electric dipole transitions in OMEGA file,
        flagged as negative numbers, see [17] for details.
        ≤ 0   Do not write the infinite-energy limits.
        > 0   Write them.

ISGPT   Integer. Default 0.
        Write partial-wave cross sections to output file
        SIGPW.DAT.
        ≤ 0   Do not write the partial collision strengths.
        > 0   Write them.

ITRMN   Integer. Default 0.
ITRMX   Integer. Default 0.
        Case ISGPT= 1, not used in case IOPT1= 0. Write in
        output file the partial-wave cross sections from
        transition ITRMN to transition ITRMX and also the
        sum of these partial wave cross sections for each
        $J\pi$ partial wave.

PRINT   Character(4). Default "FORM".
        Print style of output file OMEGA.
        "FORM"   Write the collision strengths to the
                   final output file OMEGA ASCII
                   formatted.
        Otherwise Write the collision strengths to the
                   final output file OMEGAU binary in
                   sequential access.

MZMEG   Integer. Default Huge(1)/2**23.
        Mega-words of memory available to store in
        memory the OMEM array, containing all the
        open-channel collision strengths for all energies. If
        the size of OMEM is larger than MZMEG$\times 8 \times 2^{20}$
        bytes, then a scratch file must be opened to store
        the array in hard disk. Warning, one scratch file is
        opened by each processor, if the number of
        processors is large, the hard disk can be exhausted.

MZPTS   Integer. Default 3201.
        Number of spatial grid points of the outer region,
        to perform the Numerov integration.

NAMELIST STGF MQDT variables.
    To be used only if multichannel quantum-defect theory is
activated in the calculation. These options are necessary for ICFT,
otherwise it will consume a lot of computing time unnecessarily.

IQDT   Integer 0, 1, 2. Default 0.
Controls MQDT operation. If IQDT > 0, then
higher dipole perturbing potentials will be
included in terms of variable IPERT
0   No QDT operation is performed.
1   Full MQDT, all channels treated as open, it
        uses unphysical **K**/**S** matrices.
2   Work with unphysical **K** matrices rather than
        S ones.

IMODE   Integer 0, 1, −1. Default 0.
Controls read and write of unphysical matrix
when MQDT is activated.
0   Calculates the unphysical matrix and writes it
        to output file JBIN.
1   Reads the unphysical matrix from file JBIN.
        Solution on a newly defined energy mesh
        obtained solely by interpolation of the
        previous coarse mesh data read.
-1   Single pass operation. It is performed a full
        solution on a coarse mesh and an
        interpolative solution on a fine mesh.

IJBIN   Integer. Default 0.
Write JBIN file in case IMODE=0.
0   Do not write the file.
Other-   Write the file.
wise

LMX   Integer. Default 2 (quadrupole).
If perturbations are activated, largest perturbing
multipole.

IEQ   Integer. Default -1.
Controls how often the unphysical **K** or **S** matrix
is updated.
< 0   **K** or **S** matrix is updated at every |IEQ|'th
        point of the mesh, fine for constant step in a
        previously calculated coarser energy mesh
        (IMODE=1), not so good (inefficient) for
        constant step in effective quantum number.
> 0   **K** or **S** matrix is updated at IEQ linearly
        spaced energies across the total energy range
        defined by the input energy mesh.

QETEST Float. Default 1.E−7.
Energy in scaled Rydberg units E/z^2. The **K** or **S**
matrix is only re-interpolated when the total energy
has changed by more than the QETEST value since the
last time. It gives a small time saving when using a
very fine mesh.

FNUMIN Float. Default 0.0.
The effective quantum number below which the
closed channel is omitted when IQDT= 1, 2.

FNUHYB Float. Default −1.0.
The effective quantum number below which the
closed channel is a Θ function rather than S and C
ones, when IQDT= 1, 2.

NAMELIST STGF Advanced options

We strongly recommend against the modification of these
variables, and they should only be worked by experienced
users.

IOMSW   Integer 1,0,-1. Default 1.
Controls the number of channels to be treated
as open in MQDT operation.
1   Full MQDT: it omits closed channels with
        $n < l + 0.1$.
0   Partial MQDT: keep (with $a = -a$,
        IOMIT(ICHAN)= −1 in subroutine SC when
        $a < 0$).
-1   Hybrid: use Θ functions for $\nu < $ FNUHYB and
        $\nu < l$, otherwise operate as in case IOMSW= 0.

LPRTSW Integer. Default -1 for ions and 5 for neutrals.
Value of L or $2J$ above which negative IPERT is
allowed.

ICCINT Integer 1,0. Default 1.
Whether to include closed-closed channels
perturbing integrals in both MQDT or
non-MQDT operations.
1   Include them.
0   Do not.

INTPQ   Integer 0, 1. Default 0.
Used in internal calculation subroutine.
0   use internal subroutine CORINT for closed
        channel MQDT S and C integrals.
1   use Θ function subroutines to generate Q
        integrals.

NAMELIST MESH1

Mandatory if IMESH=1. Otherwise not present.

MXE   Integer.
        Number of electron-impact energies.

EO   Float.
        First z-scaled energy of the grid in Rydberg E/z^2.

EINCR   Float.
        Energy increment step.

QNMAX   Float.
        If IQDT= 0 no MQDT except Gailitis average for
        $n >$QNMAX.

ABVTHR   Float. Default -1.0 for ions, or 1.E-3 for neutrals.
> 0 Drop energies from the grid within ABVTHR z-scaled Rydberg above the first excitation threshold.
≤ 0 Do not drop any energy.

BELTHR   Float. Default -1.0 for ions, or 1.E-3 for neutrals.
> 0 Drop energies from the grid within BELTHR z-scaled Rydberg below the first excitation threshold.
≤ 0 Do not drop any energy.
These variables are only necessary in the case of neutral targets.

For an optimum performance MXE must be an even multiplier of the number of processors. This condition is sometimes difficult to fulfill if ABVTHR or BELTHR are used, so energies are dropped from the grid. This happens commonly in the case of neutrals.

NAMELIST MESH2
Mandatory if IMESH=2 or negative, otherwise not present. This option is not yet implemented in parallel version PSTGF.

NAMELIST MESH3
Mandatory if IMESH=3, otherwise not present.

MXE   Integer.
Number of electron-impact energies. To be read afterwards as additional parameter.

Additional variables not sorted as namelist, mandatory for determined values of the variables in namelist STGF, see above.

Case IOPT1=2.
In this case, the partial waves to be calculated must be specified. It is expected several lines, each one with three integer numbers S, L, PI.

S, L, PI   Several lines of three free-format integers each.
…   List of partial waves to be calculated. These symmetries should be present in the files *H.DATXXX*.
S $2S + 1$ of the partial wave, or 0 to flag $J\pi$ coupling.
L L or $2J$ of the partial wave.
PI   Parity of the partial wave: 0 for even, 1 for odd.
A value of -1 -1 -1 or an End-Of-File marks the end of the list.

Case IOPT1=-1,-2.
In this case, the target term or level energies are treated as degenerate, and the number of degenerate terms or levels for each target energy must be read in. It is expected a first line with an integer number NASTD, and a second line with a set of NASTD integers NLEV(1:NASTD). In the case IOPT1$= -2$ the list of partial-wave symmetries is specified after the shortlist of the degeneration of the energies.

NASTD   Integer, free format.
Number of target degenerated energies.

NLEV(1:NASTD)   NASTD integer, free format.
Number of degenerated terms or levels for each energy.

Case IMESH=3.
After the namelist MESH3, expected a total of MXE floating point numbers.

EMESH(1:MXE)   MXE float, free format.
Electron-impact z-scaled energies in Rydberg, sorted from smallest to largest. For an optimum performance, MXE should be an even multiplier of the number of processors.

6.1.2. H.DAT
Mandatory, binary (see Table 1).
Binary output from the inner region codes, used as input for the outer region codes.
It can be presented in three different ways. PSTGF inquires which one is present in the following order, if several types are present, just the first one inquired as positive is used, the remaining ones are just ignored:

1. Several *H.DATXXX* files, starting by *H.DAT000* (recommended). *H.DAT000* contains the information about the target and at least the first $J\pi$ or $LS\pi$ partial wave. Each *H.DATXXX* file contains information about the channels of one (recommended) or several partial waves. In the case the number of *H.DATXXX* files is the same as the number of lines in the *sizeH.dat* file and the sorting of the *XXX* indexes agrees with the sorting of partial waves in *sizeH.dat* (see 6.1.3) the performance will be optimum.
2. One single *H.DAT* file. The single file contains the information about the target and the channels of all the partial waves calculated in the inner region. PSTGF will still work, but its performance will be not optimum because all the processors will have to read the same large file for all the symmetries.
3. One single *DSTGH.DAT* file. For inner region calculations with the old version DARC code, now obsolete. The option is kept just to allow the code work for older calculations, but it is not recommended for new work.

Information about the target, first set of records of *H.DAT000* or *H.DAT* files:

- 1: Number of electrons; nuclear charge; maximum L or $2J$ (target); maximum calculated coupling multipole; number of target terms or levels; R-matrix box size (a.u.); logarithmic derivative of the radial function in the boundary.
- 2: Target level or term energies.
- 3: Target L or $2J$ of each term or level.
- 4: Target $2S + 1$ of each term, or 0 if $J\pi$ coupling.
- 5: Coefficients for Buttle correction.
- Following $3 \times$ LRANG2 records: alternative form of the Buttle correction for DARC codes. For older versions of inner-region codes, these data could be in the different file *DBUT* instead of *H.DAT*.

Information about the $N + 1$-electron symmetries, following records of *H.DAT* or *H.DAT000* files, after the target information; or whole files *H.DATXXX* files, with $XXX \neq 000$.

- 1: L or $2J$; $2S + 1$ or 0; parity of partial wave; number of channels; number of Hamiltonian eigenvalues in partial wave; flag to check if present partial wave is the last one to be read.

Table 1
Record structure of the *H.DAT000* file.

H.DAT000, Target information	
1: 5 INT*4, 2 REAL*8	NELC,NZED,LRANG2,LAMAX,NAST,RA,BSTO
2: NAST REAL*8	(ENAT(I) I=1-NAST)
3: NAST INT*4	(LAT(I) I=1-NAST)
4: NAST INT*4	(ISAT(I) I=1-NAST)
5: 3 LRANG2 REAL*8	((COEFF(I,L) I=1-3), L=1-LRANG2)
L=1-LRANG2	
L1: 1 INT*4	NBUTD(L)
L2: NBUTD REAL*8	EBUTD(I,L) I=1-NBUTD
L3: NBUTD REAL*8	CBUTD(I,L) I=1-NBUTD
H.DAT000, First symmetry information	
1: 6 INT*4	LRGL2,NSPN2,NPTY2,NCHAN,MNP2,MORE2
2: NAST INT*4	(NCONAT(I) I=1-NAST)
3: NCHAN INT*4	(L2P(I) I=1-NCHAN)
4: LAMAX × NCHAN² REAL*8	(((CF(I,N,M) I=1-NCHAN),N=1-NCHAN),M=1-LAMAX)
5: MNP2 REAL*8	(VALUE(I) I=1-MNP2)
6: NCHAN × MNP2 REAL*8	((WMAT(I,K) K=1-NCHAN),I=1-MNP2)

Table 2
Formatted line structure of the *sizeH.dat* and *sizeBP.dat* files.

sizeH.dat	
1-NSLPI: 8X,I5,10X,I6,8X,I6,3(4X,I3)	NCHAN,NCON,MNP2,S,L,IPI
sizeBP.dat	
1-NSLPI: 3I7,9X,I3,8X,I3	NCHAN,NCON,MNP2,J2,IPI

- 2: Number of channels attached to the target level *I*.
- 3: *L* or 2*J* of the target term or level associated to channel *I*.
- 4: Coefficient of the multipole potential expansion.
- 5: Eigenvalues of the Hamiltonian, R-Matrix pole energies E_k Eq. (4).
- 6: R-Matrix amplitudes w_{ij} Eq. (4).

6.1.3. sizeH.dat

Optional, nevertheless highly recommended, ASCII (see Table 2).

All the codes that work the inner region and provide *H.DAT* files are able to provide an auxiliary formatted *sizeH.dat* file. There are two kinds of *sizeH.dat* files readable by PSTGF. It inquires if they exist in the following order:

1. *sizeH.dat* for *LSπ* or *Jπ* coupling.
2. *sizeBP.dat* for *Jπ* coupling only.

If both files exist, only *sizeH.dat* is read by PSTGF and *sizeBP.dat* is ignored. *sizeH.dat* contains the basic information about each partial wave: 2*S* + 1, *L*, *π*, number of channels, number of continuum basis × number of channels, number of eigenvalues for the *N* + 1 Hamiltonian. If 2*S* + 1 = 0 then PSTGF assumes that partial waves are presented in *Jπ* coupling and *J* = 2*L*. The number of lines in the file is interpreted by PSTGF as the number of partial waves stored in the *H.DATXXX* files (see 6.1.2). The best performance is reached if there is exactly one partial wave per *H.DATXXX* file and the sorting of the *XXX* indexes agrees with the sorting of partial waves in *sizeH.dat*

sizeBP.dat file works the same way as *sizeH.dat* file but it has no value of *S* stored, it assumes always *Jπ* coupling.

If no *sizeH.dat* or *sizeBP.dat* files are found or they have different number of lines than the number of *H.DATXXX* files, then a preliminary read of the *H.DATXXX* files has to be carried out in order to let the program know in which file is stored which partial wave, and that will take some time if the files are large. A correctly formatted *sizeH.dat* will help a good performance of PSTGF.

The file is ASCII formatted and has NSLPI lines, number of partial waves. In each partial wave is read the number of channels; the number of continuum functions (NCHAN×NRANG); number of eigenvalues of the Hamiltonian (NCON plus bound functions); 2*S* + 1 or 0; *L* or 2*J*; parity.

6.1.4. DBUT

File to store the Buttle correction. It works only for old versions of inner-region codes, currently the Buttle correction is integrated in *H.DATXXX* files.

6.2. Output files

6.2.1. Standard output

File *routf* from processor 0 and *routfgXXXX* from other processors, being *XXXX* the processor number, ASCII. Standard output, execution information, warnings and errors. If the calculation finished OK, with no errors, then the standard output files from processors different to 0 will have no additional information, and they can be safely removed to save space in the hard disk.

6.2.2. OMEGA

Final results, binary or ASCII depending on input variable PRINT.

One single file, if in the standard input the variable PRINT is specified as "FORM" then output will be formatted in an *OMEGA* file, any other value and it will be unformatted with sequential access in an *OMEGAU* file.

In any of the formats it contains the information shown in Table 3:

- 1: nuclear charge and number of electrons of the target.
- 2: Number of target terms or levels, number of impact energies and number of transitions stored, NOMWRT, it can also contain elastic transitions if variable ELAS is set to "YES".
- 3: Array containing the *L*, *S* or 0, 2*J* of each target term or level.
- 4: Array containing the term or level excitation energies of the target in Rydberg scaled units with respect to the ground state energy.

Table 3
Record or line structure of the *OMEGA* file.

OMEGA	
1: 2 INT*4	NZED NELC
2: 3 INT*4	NAST MXE NOMWRT
3: 2×NAST INT*4	(ISAT(I) LAT(I), I=1-NAST)
4: NAST REAL*8	(ENAT(I), I=1-NAST)
J=1-MXE: |NOMWRT|+1 REAL*8	E(J), (OMEGA(J,I), I=1-NOMWRT)

- $J = 1 - MXE$: Impact scaled energy in Rydberg E/z^2 and array with the Ω values for the collision strengths for that energy. The $\Omega(i-j)$ matrix is stored as the upper triangle by rows or columns depending of the sign of the variable NOMWRT.

6.2.3. SIGPW.DATXXXX
XXXX: index of processor.
ASCII, written if ISGPT= 1.
Partial wave transition amplitudes, to calculate partial cross sections.

6.2.4. JBINLS.DATXXXX
XXXX: index of processor.
Binary, written if IPRKM= 4.
Channel information to input in STGICF.

6.2.5. SMTLS.YYY.XXXX
XXXX: index of processor; *YYY*: index of symmetry.
Binary, written if IPRKM= 4 and MQDT active.
Unphysical **S** matrix, to be used as input for STGICF.

6.2.6. KMTLS.YYY.XXXX
XXXX: index of processor; *YYY*: index of symmetry.
Binary, written if IPRKM= 4 and MQDT active.
Unphysical **K** matrix, to be used as input for STGICF.

These two files should be used with care, as the output is sorted by each processor in terms of its local energy grid. ICFT code does not work with present implementation of PSTGF, so these files cannot be used unless PSTGICF is updated in a consistent way of current PSTGF version.

6.2.7. OMEGDR
ASCII, written if NDRMET> 0. Same format of *OMEGA* file, replacing the number of transitions NOMWRT for initial states for DR NAST.

Dielectronic recombination cross sections. PSTGF is not the appropriate code to use to calculate DR. The radiation-damped code PSTGFDAMP should be used instead.

6.3. STRENGTH.DAT

ASCII, written if IDIP= 1.
E1 line strengths *S* between all target terms or levels.

6.3.1. TERM.DAT
ASCII, written if IPRKM= 4.
Information about the target terms or level: $2S + 1$, *L*, π and *E*.

6.4. Scratch files

6.4.1. SCRATCH1
File to store the OMEM array if its dimension is larger than MZMEG×2^{20}. In this case a scratch file is opened and the array OMEM is stored on the hard disk. One file is opened by each processor, if the number of processors is large, caution must be taken to ensure that hard disk is not unexpectedly filled.

The best practice is not to modify the default value of MZMEG to ensure that no scratch files are open. There is also a reduction the input and the output time.

7. Test cases

We have used v1.1 to calculate the electron-impact excitation of the ion Ni^{3+}, for 42 $J\pi$ partial waves, with a maximum of 1818 channels per partial wave and 10 energies per processor. We compare the computation time with the version v0.87. Fig. 3 shows the real nature of the problem, how the calculation time is very different for each value of the impact energy. The difference between the named slow and fast energies is self-evident, reaching in the worst cases a factor ten. Most of the calculation time will be spent on energies around the 150 index.

Fig. 4 illustrates the workload unbalance versus processor time. When some processors have to calculate a larger number of slow energies, other processors do not. In the slowest cases, the relative difference in the processing time can reach 20%, this is unused processing time. In contrast, twisting the distribution of energy points and forcing all processors to work all energies, the workload balancing is greatly improved reducing the overall processing time by approximately 15%, and the maximum processor waiting time to less than one half. Both curves in Fig. 4 have the same area, but curve for v1.1 is flatter and its maximum (total calculation time) is a 15% smaller.

This difference is more evident in a more extreme calculation case. Fig. 5 shows a calculation case for neutral Fe, with 10 partial waves and number of channels between 6943 and 7102. In this case we perform the calculation in the most extreme case, one processor per energy, and equal number of partial waves than energies, so each processor works only one symmetry in each energy. The unequal workload balance is more extreme in this case, reaching a comparative factor of 4 in the worst case. The effect of the redistribution of the energies and symmetries among the processors leads to a reduction of a factor 1/2 in the computing time.

8. Example of input

We show an example the ASCII standard input (*dstgf*) for the case of electron-impact excitation of ion Ge^{2+}, 30-electron system plus projectile.

For the inner region calculation, we carried out 42 relativistic partial waves, with $J = \frac{1}{2}$ to $20\frac{1}{2}$, even and odd parity. From the inner region, 42 *H.DATXXX* files were produced and one *sizeBP.dat* of length 42 lines, transcribing the partial wave symmetry associated with *H.DATXXX* file. Section 6.1.3.

We calculate in the outer region the collision strengths for the first 40 partial waves, from the first excitation threshold of the ion ($0.56\,\mathrm{Ry} = 0.14z^2$) to twice the ionisation limit ($5.0\,\mathrm{Ry} = 1.25z^2$), with a fine energy mesh of $10^{-5}z^2$. It is a good practice to start the energy grid a little below the first threshold, so the user can double-check that all collision strengths are zero below such threshold. This is useful not only as first check for the validity of the calculation, but also for postprocessing tasks, in case an interpolation or spline of the collision strengths below the threshold is carried out. In the case of neutral targets, the drop of the cross section below the threshold is smoother than in the case of ions, so it is possible to have non-zero cross sections for energies slightly below such threshold.
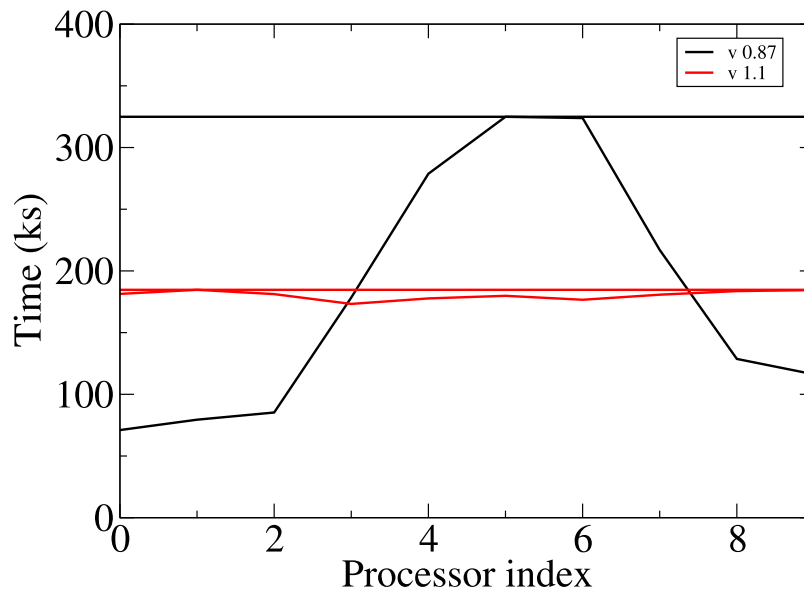
Fig. 3. Total processor time spent in each energy Ni^{3+}. The calculation of the transition matrix S turns much slower when the number of open channels n_o and closed channels n_c is similar.

Fig. 4. Total time of each processor Ni^{3+}. The horizontal lines mark the total calculation time (maximum).

```
&STGF IMESH=1 IQDT=0 IPRINT=-2 IOPT1=2 LRGLAM=-1 &END      0 17 0
&MESH1 MXE=120000 E0=0.13000 EINCR=0.00001 &END            0 17 1
0 1 0                                                      0 19 0
0 1 1                                                      0 19 1
0 3 0                                                      0 21 0
0 3 1                                                      0 21 1
0 5 0                                                      0 23 0
0 5 1                                                      0 23 1
0 7 0                                                      0 25 0
0 7 1                                                      0 25 1
0 9 0                                                      0 27 0
0 9 1                                                      0 27 1
0 11 0                                                     0 29 0
0 11 1                                                     0 29 1
0 13 0                                                     0 31 0
0 13 1                                                     0 31 1
0 15 0                                                     0 33 0
0 15 1                                                     0 33 1
```

Fig. 5. Total time of each processor Fe neutral. The horizontal lines mark the total calculation time (maximum).

```
0 35 0
0 35 1
0 37 0
0 37 1
0 39 0
0 39 1
-1 -1 -1
```

With this input we will perform the calculation of the first 40 partial waves specified in *sizeBP.dat* file and no top-up, with the full balancing efficiency of PSTGF, distributing read files in the different processors and doing the efficient energy split in processors and partial waves.

In order to use this efficient split, we are forced to deactivate the top-up procedure, through setting *LRGLAM* to negative. As a second step we perform the high-*J* top-up, in this case it is necessary that all processors calculate the collision strengths for the same energies at least for the two last partial waves. Hence, to perform the top-up, we carry out a second calculation including just these two last partial waves. In this second calculation, we do not take advantage of the efficiency improvements of v1.1-2018, but it is a small calculation, so the loss in user time is minimal.

```
&STGF IMESH=1 IQDT=0 IPRINT=-2 IOPT1=2 LRGLAM=41 &END
&MESH1 MXE=1200 E0=0.130 EINCR=0.001 &END
0 41 0
0 41 1
-1 -1 -1
```

This time, we set the top-up variable *LRGLAM* to the value of 2*J* of the last partial wave calculated = 41. The cross sections for the higher partial waves do not present resonant structure, so it is not necessary to calculate an energy mesh as fine as in the previous calculation. With the simple post-processing tool OMADD, we can add-up the two *OMEGA* files from the two calculations.

9. Future work

All these improvements in the implementation have been applied to the undamped version of PSTGF. Nevertheless, they do not modify the physics of the problem. As a consequence all these modifications can be integrated within the radiationally damped version PSTGFDAMP [18] with a similar increase of efficiency.

Likewise, it would be necessary to modify the ICFT code PST-GICF [7] to allow for the reading of the files containing the **K** and **S** in a consistent way with current version of PSTGF.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Finally, we acknowledge once more the first version of STGF written by Professor M.J. Seaton, a foundation for the current work of several people presented here.

References

[1] P.G. Burke, R-Matrix of Atomic Collisions: Application To Atomic, Molecular, and Optical Processes, Springer-Verlag, New-York, 2011.
[2] V. Burke, P. Burke, N. Scott, Comput. Phys. Comm. 69 (1) (1992) 76–98, http://dx.doi.org/10.1016/0010-4655(92)90131-H.
[3] K.A. Berrington, W.B. Eissner, P.H. Norrington, Comput. Phys. Comm. 92 (1995) 290.
[4] P.H. Norrington, I.P. Grant, J. Phys. B: At. Mol. Phys. 14 (1981) L261–L267.
[5] P.H. Norrington, I.P. Grant, J. Phys. B: At. Mol. Opt. Phys. 20 (1987) 4869–4881.
[6] N.R. Badnell, J. Phys. B: At. Mol. Opt. Phys. 32 (23) (1999) 5583, http://dx.doi.org/10.1088/0953-4075/32/23/312.
[7] D.C. Griffin, N.R. Badnell, M.S. Pindzola, J. Phys. B: At. Mol. Opt. Phys. 31 (16) (1998) 3713–3727, http://dx.doi.org/10.1088/0953-4075/31/16/022.
[8] T.W. Gorczyca, M.S. Pindzola, F.S. Shieh, C.L. McCreary, Comput. Phys. Comm. 88 (2) (1995) 211, http://dx.doi.org/10.1016/0010-4655(95)00068-Q.
[9] N.R. Badnell, Comput. Phys. Comm. 182 (7) (2011) 1528–1535, http://dx.doi.org/10.1016/j.cpc.2011.03.023.
[10] C. Froese-Fischer, G. Tachiev, G. Gaigalas, M.R. Godefroid, Comput. Phys. Comm. 176 (8) (2007) 559–579, http://dx.doi.org/10.1016/j.cpc.2007.01.006.
[11] K.G. Dyall, I.P. Grant, C.T. Johnson, F.A. Parpia, E.P. Plummer, Comput. Phys. Comm. 55 (1989) 425.
[12] F.A. Parpia, C.F. Fischer, I.P. Grant, Comput. Phys. Comm. 94 (1996) 249.
[13] A. Hibbert, Comput. Phys. Comm. 9 (3) (1975) 141–172, http://dx.doi.org/10.1016/0010-4655(75)90103-4.

[14] P.G. Burke, A. Hibbert, W.D. Robb, J. Phys. B: At. Mol. Phys. 4 (2) (1971) 153, http://dx.doi.org/10.1088/0022-3700/4/2/002.

[15] K.A. Berrington, P.G. Burke, K. Butler, M.J. Seaton, P.J. Storey, K.T. Taylor, Y. Yan, J. Phys. B: At. Mol. Phys. 20 (23) (1987) 6379, http://dx.doi.org/10.1088/0022-3700/20/23/027.

[16] O. Zatsarinny, Comput. Phys. Comm. 174 (2006) 273–356, http://dx.doi.org/10.1016/j.cpc.2005.10.006.

[17] A. Burgess, J.A. Tully, Astron. Astrophys. 254 (1992) 436–453.

[18] A.D. Whiteford, N.R. Badnell, C.P. Ballance, M.G. O'Mullane, H.P. Summers, A.L. Thomas, J. Phys. B 34 (15) (2001) 3179–3191, http://dx.doi.org/10.1088/0953-4075/34/15/320.