Users Manual for the Parallel LS/Breit-Pauli/DARC R-Matrix Programs

Authors :

C P Ballance[1] and D C Griffin[1]

[1]Department of Physics, Rollins College, Florida, FL 32789 USA

Collaborators :

N R Badnell[2]
D M Mitnik[3]
K A Berrington[4]

[2]Department of Physics, University of Strathclyde, Glasgow G4 0NG, UK
[3]Departamento de Fsica, FCEN, Universidad de Buenos Aires, and Instituto de Astronoma y Fsica del Espacio (IAFE), Casilla de Correo 67, Sucursal 28, (C1428EGA) Buenos Aires, Argentina
[4]School of Science and Mathematics, Sheffield Hallam University, Sheffield S1 1WB, UK

Abstract.

This document describes the use of the parallel R-matrix codes developed by the authors from modified serial versions of the RMATRX1 suite of programs (Berrington *et al* 1995 *Comput. Phys. Commun.* **92** 290), the non-exchange R-matrix programs (Burke *et al* 1992 *Comput. Phys. Commun.* **69** 76), and the Dirac Atomic R-matrix Code (DARC) (P. H. Norrington 2004 http://www.am.qub.ac.uk/DARC). This description assumes the availability of a Fortran90 Compiler and MPI, ScaLAPACK, BLAS, and BLACS Libraries. Although these programs will run on a variety of parallel computer platforms, the examples shown are from calculations performed at the National Energy Research Scientific Computing Center at Lawrence Berkeley National Laboratory, the Center for Computational Sciences at Oak Ridge National Laboratory, and on a 26 processor Opteron computer cluster at Rollins College.

1. Contents

2. History

The parallel R-matrix codes described here were developed from modified versions of the serial RMATRX1 codes [1], the non-exchange R-matrix programs [2] and the Dirac Atomic R-matrix Code (DARC) [3]. The development of the original serial programs were primarily due to the efforts of physicists at the Queen's University of Belfast. Since then, there have been substantial improvements in the serial versions of these codes by physicists at Strathclyde University, Rollins College, and members of the Iron/Opacity/RmaX projects.

From the late nineties onward, a Rollins/Strathclyde/Auburn collaboration began to develop a fully parallel set of LS/Breit-Pauli (BP) codes with options for the inclusion of pseudostates. This allowed us to overcome computational bottlenecks that restricted calculations performed using the serial programs to $(N+1)$-electron Hamiltonian matrices of order 10,000 [4]. We now routinely handle cases with Hamiltonian matrices of order 30,000 to 40,000 and the largest matrix to date is of order 50,000. More recently, a program has been added to the DARC suite of codes to enable it to work with our parallel LS/BP diagonalization and asymptotic programs, and we have parallelized several of the DARC codes. This enables us to handle much larger calculations for highly ionized species, where a fully relativistic description in the inner region is more important.

The primary goal of the current effort has been to provide sets of parallel excitation/ionization codes, based on the R-matrix method, that can be employed to make accurate large-scale scattering calculations on a wide variety of atomic systems from neutral and near neutral species, where coupling of the bound states to the target continuum is important, to highly ionized species, where relativistic effects are essential. This is an ongoing project and changes are constantly being made to improve on both the versatility and efficiency of these programs. Although the codes are appropriate for a variety of species, the number of states that sometimes must be included in order to obtain accurate results, especially for neutral or near-neutral complex atomic systems, can render the calculation impractical, even on the current generation of massively parallel computers.

3. An overview of the codes

3.1. The LS/Breit-Pauli codes

As with the original serial version of the RMATRX1 package, the parallel version of the LS/Breit-Pauli (BP) R-matrix package consists of a series of programs that can be used to perform either LS or BP intermediate-coupling scattering calculations. A summary of recent modifications made to these parallel LS/BP codes is given in Appendix A. The target states may be generated using the programs AUTOSTRUCTURE, CIV3 or MCHF, but the use of these programs will not be described here.

- **pstg1r.f** – reads the bound state radial orbitals from a structure calculation, generates the orbital basis for the $(N+1)$-electron continuum, and calculates all radial integrals. The calculations of the radial integrals are distributed over the processors.
- **pstg2r.f** – carries out the angular algebra calculations in LS coupling and generates the $(N+1)$-electron matrix elements in the LS representation. The calculations are distributed over the processors by $LS\Pi$ partial wave.
- **pstg2.5.f** – (for a BP calculation only) writes a series of files containing the $(N+1)$-electron matrix elements for all the $LS\Pi$ partial waves needed to form the matrix elements in intermediate coupling for a given range of $J\Pi$. The writing of the individual files is distributed over the processors.
- **precupd.f** – (for a BP calculation only) generates term-coupling coefficients for the N-electron target states and transforms the $(N+1)$-electron matrix elements from LS to jK coupling. The calculations are distributed over the processors by $J\Pi$ partial wave. **This code is also referred to as pstgjk.f; precupd.f and pstgjk.f are identical codes.**
- **pstg3r.f** – reads the information from pstg2r.f for an LS run or the information from precupd.f/pstgjk.f for a BP run, forms the $(N+1)$-electron Hamiltonian matrices, and diagonalizes them to generate the surface amplitudes and R-matrix poles. The diagonalization of the Hamiltonian matrices is distributed over the various processors.
- **pstgf.f** – solves the coupled equations in the external region using perturbative methods and matches to the R-matrix on the inner-region boundary to generate collision strengths. The calculation is distributed over processor by electron energy.
- **stglib.f** – a set of subroutines called by pstg2r.f and precupd.f

In addition, there is the parallel asymptotic code **pstgicf.f**. It reads unphysical K-matrices or S-matrices in LS coupling, generated using the Multi-channel Quantum Defect Theory (MQDT) options in pstgf.f, transforms them to intermediate coupling, and uses MQDT to generate collision strengths in intermediate coupling. This program can be used to calculate excitation cross sections in intermediate coupling for ions in intermediate stages of ionization that agree well with the results obtained from a full BP calculation. However, we will not discuss such a calculation in this document.

3.2. The non-exchange R-matrix codes

The non-exchange R-matrix programs consist of three inner region codes to perform LS coupling R-matrix calculations without electron exchange. These are especially useful for determining the contributions from the high partial waves. Of these, only two stages (pstg2nx.f and pstg3nx.f) have been parallelized. The outer region portion of a non-exchange calculation is performed using pstgf.f. A summary of recent changes made to the parallel non-exchange codes is given in Appendix B.

- **stg1nx.f** – determines the angular coefficients for a non-exchange calculation in LS coupling. It is very fast and has not been parallelized.
- **pstg2nx.f** – generates the orbital basis for the $(N+1)$-electron continuum, calculates all non-exchange radial integrals and forms the $(N+1)$-electron non-exchange matrix elements. The calculations are distributed over the processors by the spin of the target and the total orbital angular momentum and parity of the $(N+1)$-electron system.
- **pstg3nx.f** – reads the information from pstg2nx.f, forms the $(N+1)$-electron Hamiltonian matrices without exchange, and diagonalizes them to generate the surface amplitudes and R-matrix poles. The diagonalization of the Hamiltonian matrices is distributed over the various processors.

3.3. The Dirac R-matrix codes (DARC)

The Dirac R-matrix codes perform relativistic scattering calculations in the inner region, but use a non-relativistic approximation in the asymptotic region. Our parallel version of DARC allows us to employ our parallel diagonalization code (pstg3r.f) and our parallel asymptotic code (pstgf.f) to complete a DARC calculation. A summary of the recent changes made to the parallel DARC codes is given Appendix C. The target states for a DARC run are generated using the code GRASP. Since many users of the R-matrix codes are not as familiar with this code as they are with other structure programs, our example of a DARC run will include input for GRASP.

- **stg0d.f** – converts bound-state orbital data output by GRASP into a format that can be read by the scattering codes. It is very fast and has not been parallelized.
- **stg1d_orb.f** – generates the orbital basis for the $(N+1)$-electron continuum. It is very fast and has not been parallelized.
- **stg1d_int.f** – calculates the radial integrals. It is fast and has not been parallelized.

- **pstg2d.f** – calculates the angular coefficients and forms the $(N + 1)$-electron matrix elements. The calculations are distributed over the various processors by $J\Pi$ partial wave.
- **pdto3.f** – The parallel version of the code dto3.f originally written by K. A. Berrington, reads the $(N + 1)$-electron matrix elements generated by pstg2d.f and writes them in a form that can be read by the parallel diagonalization code pstg3r.f. The calculations are distributed over processor by $J\Pi$ partial wave.

3.4. Essential utility codes

There are currently dozens of utility codes used to manipulate or interrogate the large binary passing files, or to extract information from output files. Below, we describe just three utility codes that we believe are essential. The other utility programs are also available; each contains a brief statement of its purpose, as well as a brief description of its required input, as comments near the top of the program.

- **arrange.f -** pstgf.f writes collision strength files **OMEGAXXX**, where XXX stands for a number from 000 up to a maximum of 999. Each file contains collision strengths for a group of incident electron energies. However, in order to achieve the best load balance between processors in pstgf.f, the energy points are **not** written in monotonically increasing energy order. arrange.f creates a single energy ordered file **OMEGAZ** from all OMEGAXXX files. Furthermore, if the narrow resonances are not sufficiently resolved from a given run, it can be used to interweave the collision strengths for extra energy points created in subsequent runs. No input is required for this program.
- **omadd.f -** This utility code has two primary purposes:
  - The first is to remove isolated numerical failures and/or unresolved resonances from a large **OMEGA** file.
  - The second is to add the values of the collision strengths from two OMEGA files. For example, in a LS scattering calculation, when the LS/BP R-matrix codes are used to determine the low partial-wave contributions with exchange and the non-exchange codes are used to generate the high partial-wave contributions, this program can be used to add the collision strengths arising from the two runs. omadd.f interpolates the collision strengths from the coarse energy mesh used with the non-exchange run onto the fine energy mesh used in the exchange calculation and then adds the two sets of collision strengths.

  input variables for omadd.f are given at the top of the code.

- **stgsig.f -** it extracts collision strengths or calculates cross sections, as a function of incident electron energy, for a set of specified transitions from the OMEGA file. The input variables are explained at the top of the code.

4. Obtaining the codes

The programs summarized above, plus the Dirac structure code GRASP and a number of other useful utility codes are available from the following two sites.

- Rollins College : http://vanadium.rollins.edu/codes
- Strathclyde University : http://amdpp.phys.strath.ac.uk/tamoc/code.html

4.1. Architectures and compilers

There are both serial and parallel versions of the *R*-matrix codes. The codes are written in fortran with the parallelism implemented through the Message Passing Interface (MPI). The current philosophy is that the serial codes should maintain their fortran77 roots, and be capable of running on a linux box with the free GNU 77 compiler. However, the parallel codes described in the present document include many fortran90 features in order to allow for increased efficiency and reduce memory requirements. The parallel codes have been tested on a variety of parallel platforms, but primarily on:

- An Opteron computer cluster with a Portland Group compiler.
- Two IBM SP series: (1) Seaborg at NERSC and (2) Cheetah at ORNL with IBM compilers.
- An SGI ALtix parallel system with an Intel compiler.

4.2. Warnings

- You should NEVER mix other versions of these codes with the current release. They may not be up-to-date on bug fixes and there may be inconsistencies in the format employed in the passing files.
- NEVER mix serial and parallel runs within the same directory. There are differences in the required passing files that can cause failures.

5. Compiling the codes

The compile examples will be restricted to the IBM SP and Opteron machines. Examples of the compilation of the utility codes are not included here; these codes are relatively simple and they all take no more than a few minutes to run, making compiler flags unnecessary.

All programs except for the utility codes contain INCLUDE statements so that dimensions can be specified in a separate file. The LS/BP inner-region codes (pstg1r.f, pstg2r.f, pstg2.5.f, pstgjk.f, and pstg3r.f) require one PARAM file, the non-exchange inner-region codes (stg1nx.f, pstg2nx.f, and pstg3nx.f) require a second PARAM file, the inner-region DARC codes (stg1d_orb.f, stg1d_int.f, and pstg2d.f) require a third PARAM file, pdto3.f requires the same PARAM files as the inner-region LS/BP codes, and finally the asymptotic code pstgf.f requires a fourth PARAM file. The data in these files set the dimensions of large arrays and ultimately the memory requirements of the compiled codes; do not forget to include these files within the directory where the corresponding programs are to be compiled. It is advised that you name the four PARAM files as PARAM.bp, PARAM.nx, PARAM.darc, and PARAM.f and then copy them to PARAM before compiling the appropriate codes. Sample PARAM files are shown in Appendix D.

5.1. IBM SP series

For those who have run on the IBM SP series of machines, you may be familiar with compilations using the 32 bit default implementation and the **bmaxdata** and **bmaxstack** options. However, these are not recommended for compilation of the parallel codes. With the implementation of a 64 bit MPI library, the **bmaxdata** and **bmaxstack** options are no longer needed and continued use of them can lead to such errors as segmentation faults. All IBM systems require the use of Loadleveler to submit batch jobs. This will be illustrated in several of the examples covered in Section 6.

5.1.1. LS/BP codes The codes use only MPI libraries to implement parallelism, but these libraries have been bundled with codes that employ OPEN_MP; therefore, one must use mpxlf90_r instead of mpxlf90 for compilation on the IBM SP. Only pstg2r.f and precupd.f require the library routines contained within stglib.f. The inner region LS/BP codes should be compiled as follows:

- mpxlf90_r -qfixed=72 pstg1r.f -o pstg1r.x -O3 -q64 -qstrict
- mpxlf90_r -qfixed=72 pstg2r.f stglib.f -o pstg2r.x -O3 -q64
- mpxlf90_r -qfixed=72 pstg2.5.f -o pstg2.5.x -O3 -q64
- mpxlf90_r -qfixed=72 precupd.f stglib.f -o precupd.x -O3 -q64

pstg3r.f requires additional library linkage.

- At ORNL, the 64 bit MPI is implemented by the 64 bit switch (i.e., -q64):
  - mpxlf90_r -qfixed=72 -lblas -L/usr/local/lib64 -lscalapack -lblacsF77init -lblacs -lessl pstg3r.f -o pstg3r.x -O4 -q64
- At NERSC:
  - 1. module add scalapack_64
  - 2. mpxlf90_r -qfixed=72 pstg3r.f -o pstg3r.x $BLACS $PBLAS $SCALAPACK -lessl -O4 -q64

Finally, the standard version of pstgf.f uses the accelerated routine DGEMM which can have a large impact on speed. However, it requires that the Engineering and Scientific Subroutine Library (essl) be linked in the compile statement as shown below:

- mpxlf90_r -qfixed=72 pstgf.f -o pstgf.x -O3 -q64 -lessl -qstrict

There is a version of pstgf.f without this option and it has been named **pstgf.m2.f**. It can be compiled without the lessl library as follows:

- mpxlf90_r -qfixed=72 pstgf.m2.f -o pstgf.m2.x -O3 -q64 -qstrict

5.1.2. Non-exchange codes The non-exchange inner region codes should be compiled with:

- xlf90_r -qfixed=72 stg1nx.f -o stg1nx.x -O3 -q64 -qstrict
- mpxlf90_r -qfixed=72 pstg2nx.f -o pstg2nx.x -O3 -q64
- At ORNL
  - mpxlf90_r -qfixed=72 -lblas -L/usr/local/lib64 -lscalapack -lblacsF77init -lblacs -lessl pstg3nx.f -o pstg3nx.x -O4 -q64
- At NERSC:
  - 1. module add scalapack_64
  - 2. mpxlf90_r -qfixed=72 pstg3nx.f -o pstg3nx.x $BLACS $PBLAS $SCALAPACK -lessl -O4 -q64

5.1.3. DARC codes The Dirac structure code grasp.f and the DARC suite of inner region codes should be compiled as follows:

- xlf90_r -qfixed=72 grasp.f lapackblas.f njgraf.f -o grasp.x -q64 -qstrict
- xlf90_r -qfixed=72 stg1d0.f -o stg1d0.x -q64 -qstrict
- xlf90_r -qfixed=72 stg1d_orbs.f lapackblas.f -o stg1d_orbs.x -q64 -qstrict

- xlf90_r -qfixed=72 stg1d_ints.f -o stg1d_ints.x -q64 -qstrict -O3
- mpxlf90_r -qfixed=72 pstg2d.f lapackblas.f -o pstg2d.x -qhot -q64
- mpxlf90_r -qfixed=72 pdto3.f -o pdto3.x -qhot -q64

Of course, pstg3r.f and pstgf.f are required for a full DARC run and they should be compiled as shown on the previous page.

5.2. Beowulf Opteron Cluster : Portland Group compiler

The Rollins College Opteron cluster employs 64 bit processors. All these codes have been tested on the Opteron using the Portland Group compiler. Other fortran90 compilers should work equally well, but will require testing.

5.2.1. LS/BP codes The *LS*/BP inner region codes should be compiled as follows.

- mpif90 pstg1r.f -o pstg1r.x -fast -Kieee
- mpif90 pstg2r.f stglib.f -o pstg2r.x -fast -Kieee
- mpif90 precupd.f stglib.f -o precupd.x -fast -Kieee
- mpif90 pstg2.5.f -o pstg2.5.x -fast -Kieee

pstg3r.f requires ScaLapack in conjunction with the ATLAS library versions of BLAS and lapack.

- mpif90 -tp=k8-64 -Mcache_align pstg3r.f -o pstg3r.x -L /opt/acml2.5.0/pgi64/lib -lscalapack -lblacsF77init -lblacs -lacml -Kieee -fastsse

As mentioned above, the standard version of pstgf.f employs the accelerated routine DGEMM which can have a large impact on speed. This version of the code is compiled as:

- mpif90 pstgf.f -o pstgf.x -tp=k8-64 -Mcache_align -L/opt/acml2.5.0/pgi64/lib -lacml -fast -Kieee

pstgf.m2.f is the version of the code that does not employ this option and it can be compiled more simply using the command

- mpif90 pstgf.m2.f -o pstgf.m2.x -fast -Kieee

5.2.2. Non-exchange codes The inner region non-exchange codes should be compiled with :

- pgf90 stg1nx.f -o stg1nx.x -fast -Kieee
- mpif90 pstg2nx.f -o pstg2nx.x -fast -Kieee
- mpif90 -tp=k8-64 -Mcache_align pstg3nx.f -o pstg3nx.x -L /opt/acml2.5.0/pgi64/lib -lscalapack -lblacsF77init -lblacs -lacml -Kieee -fastsse

5.2.3. DARC codes The Dirac structure code grasp.f and the DARC suite of inner region codes should be compiled as follows:

- pgf90 grasp.f lapackblas.f njgraf.f -o grasp.x
- pgf90 stg1d0.f -o stg1d0.x
- pgf90 stg1d_orb.f lapackblas.f -o stg1d_orb.x -fastsse
- pgf90 stg1d_int.f -o stg1d_int.x -fastsse
- mpif90 pstg2d.f lapackblas.f -o pstg2d.x -fastsse
- mpif90 pdto3.f -o pdto3.x -fast

Of course, pstg3r.f and pstgf.f are required for a full DARC run and they should be compiled as shown above.

6. An *LS* *R*-matrix with pseudostates example: hydrogen

As an example, we will now consider an *R*-matrix with pseudo states (RMPS) calculation for the excitation of hydrogen, run on the 26 processor Opteron cluster at Rollins College. Before a scattering calculation can begin, one must generate the target states. In general, these orbitals may be determined using atomic structure packages such as AUTOSTRUCTURE, MCHF, or CIV3. In this case, we used AUTOSTRUCTURE (available at amdpp.phys.strath.ac.uk/autos/) to generate both the spectroscopic states and the Laguerre pseudo states used in this RMPS calculation. These orbitals are written to a formatted file called **radial** that is required by the scattering codes. The flow chart for the *LS*/BP exchange codes for an *LS*-coupling *R*-matrix calculation such as this RMPS calculation for H is shown in figure 1.

6.1. pstg1r.f

The only input file required by **pstg1r.f**, other than radial, is **dstg1**. The **dstg1** file for this run is shown below:

```
S.S. H RMPS Calculation
&STG1A ISMITN=1 /
&STG1B MAXLA=5 MAXLT=13 MAXC=75 MAXE=6 NMIN=5 NMAX=12
LMIN=0 LMAX=5 /
```

Note that we have employed a forward slash to end the namelist input. This works with both the IBM and Portland Group compilers. However, with some compilers, one must employ &END to complete namelist input. Also note that the only required

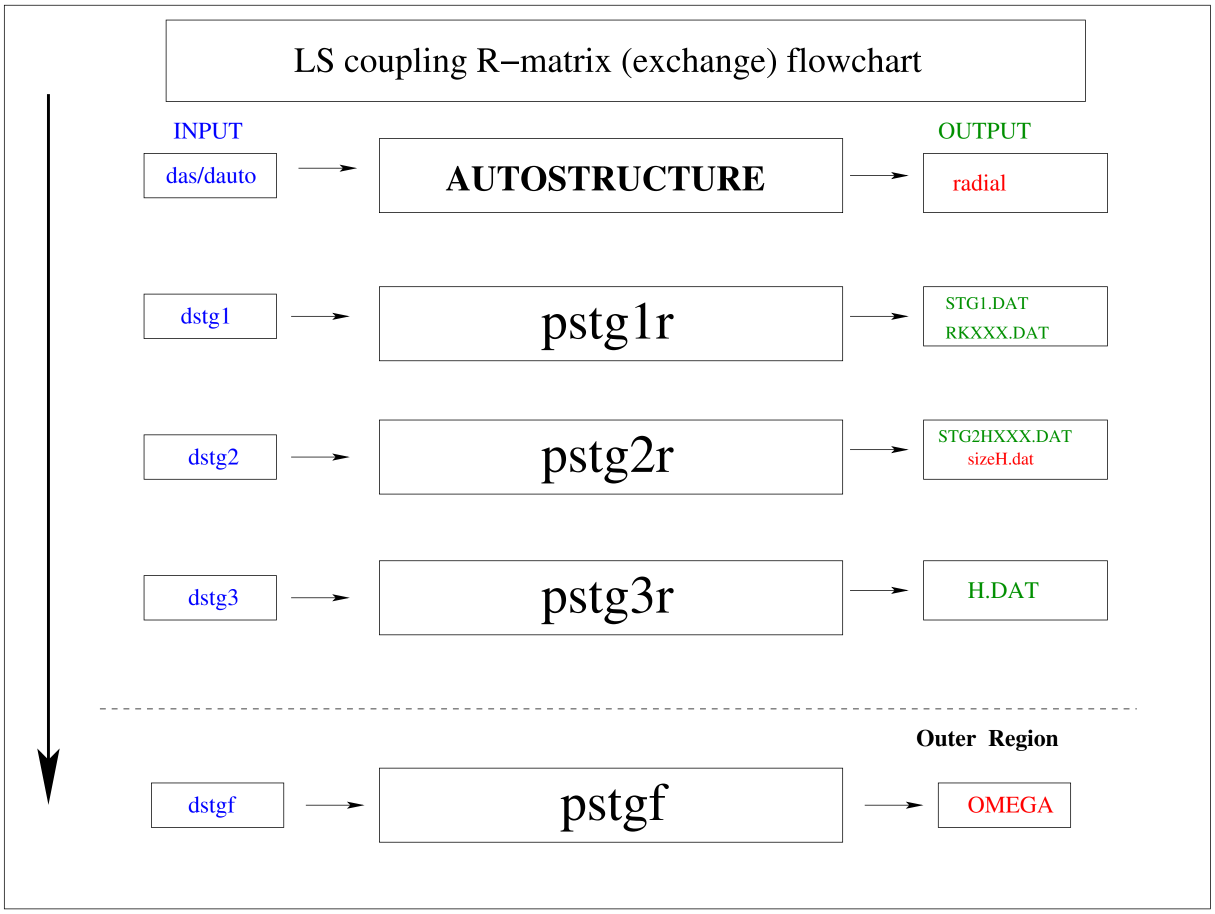


Figure 1. Flowchart of the *LS R*-matrix inner-region codes. The output files in green are binary files and the ones in red are formatted files. Note that there is an option to print the OMEGA file as a binary file, rather than a formatted file.

input on the first line is S.S., which indicates that the target states are to be read in as numerical orbitals from radial in SUPERSTRUCTURE format (STO is used for CIV3 target-state input). The remainder of the first line can be used to identify the run.

- ISMITN = 1, Indicates that a pseudostate calculation is being carried out. Pseudostates are being used to represent the high Rydberg states and the target continuum.
- MAXLA = maximum value of the total angular momentum for the target terms (in this case we have included spectroscopic and pseudo states up to L = 5).
- MAXLT = maximum value of the total angular momentum of (N+1)-electron

system (in this case we are doing a full exchange calculation for all partial waves up to $L = 13$).

- MAXC = maximum number of the $(N + 1)$-electron continuum basis orbitals per angular momentum.
- MAXE (optional) if specified, is the maximum scattering energy in rydbergs for your problem. If the MAXC you have specified is too small for this MAXE, an estimate of the value required is printed and execution halts.

In general, for a given MAXC, the maximum scattering energy for which accurate collision strengths can be calculated is approximately half the maximum eigen-energy of the continuum basis orbitals. However, for a number of cases, we have made calculations at higher energies than this without serious problems. However, if this is done, the collision strengths should be monitored carefully at high energies for oscillations.

The variables NMIN, NMAX, LMIN, LMAX are used only in a calculation that includes target pseudostates:

- NMIN = minimum n value for the pseudostates.
- NMAX = maximum n value for the pseudostates.
- LMIN = minimum l value for the pseudostates.
- LMAX = maximum l value for the pseudostates.

In this calculation, we are employing Laguerre pseudostates from 5s to 12g to represent the high Rydberg states and the target continuum.

pstg1r.f is executed on 16 processors on the Opteron cluster using the command:

```
nohup time mpirun -np 16 pstg1r.x &
```

Since the code is being run in the background, we have used the nohup command so that the system will write information regarding the run to the output file nohup.out. The time command also writes timing information to the nohup.out file.

6.2. pstg2r.f

Other than the passing files, the only required input deck for **pstg2r.f** is **dstg2**. The **dstg2** file for this run is shown on the next page.

```
S.S. H RMPS Calculation
&STG2A ISORT=1 /
&STG2B MAXORB=57 NELC=1 NAST=57 INAST=0 MINLT=0 MAXLT=13
MINST=1 MAXST=3 /
1 0 2 0 2 1 3 0 3 1 3 2 4 0 4 1 4 2 4 3 5 0 5 1 5 2 5 3 5 4
6 0 6 1 6 2 6 3 6 4 6 5 7 0 7 1 7 2 7 3 7 4 7 5 8 0 8 1 8 2
8 3 8 4 8 5 9 0 9 1 9 2 9 3 9 4 9 5 10 0 10 1 10 2 10 3 10 4 10 5
11 0 11 1 11 2 11 3 11 4 11 5 12 0 12 1 12 2 12 3 12 4 12 5
1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1
2 0 0
2 0 0
2 1 1
2 0 0
... skipping terms
2 5 1
1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2
2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 2
```

Again, in the first card, only S.S. is required. The rest of the line can be used to identify the run

- ISORT = 1 causes the code to consider all N-electron terms of the same $SL\Pi$ symmetry together, regardless of the order in which they are input. This improves the efficiency of the angular algebra calculations. However, it assumes that the number of terms to be included in the close-coupling expansion of target is equal to the number of all possible terms derived from the listed configurations. If ISORT = 0, then one can specify any number of terms to be included in the close-coupling expansion of the target, up to the maximum number of possible terms; however, for maximum efficiency it is then better to group them by $SL\Pi$.

- MAXORB = number of orbitals that will be used to define the target configurations.
- NELC = number of target electrons.
- NAST = number of terms to be included in the close-coupling expansion of the target.
- INAST = number of $(N+1)$-electron $SL\Pi$ symmetries to be included in the calculation. If INAST = 0, then the code generates **all possible** $(N+1)$-electron $SL\Pi$ symmetries internally according to the ranges specified by MINLT, MAXLT, MINST, and MAXST below. If INAST is not equal to zero, then the INAST symmetries must be entered explicitly at the end of the input file in the form $2S+1\ L\ \Pi$, where $\Pi = 0$ for even parity and $\Pi = 1$ for odd parity.
- MINLT = minimum value of L.
- MAXLT = maximum value of L.
- MINST = minimum value of $2S+1$.
- MAXST = maximum value of $2S+1$.

Following all namelist input, one enters the following data in free-format form:

- The nl values for all target orbitals, including any pseudo-orbitals. In this case, we are including spectroscopic and pseudo-orbitals from 1s to 12h.
- The number of base configurations from which all N-electron configurations are to be generated.
- The minimum occupation numbers for all N-electron configurations.
- The maximum occupation numbers for all N-electron configurations.
- A list of base configurations from which all N-electron configurations can be specified. The last number in this list is the maximum number of promotions from the base configuration to be allowed within the constraints specified by the minimum and maximum occupation numbers. For this simple case, the code will generate all configurations from 1s to 12h. For more complex cases, it is safer to input all configurations explicitly, each one with zero promotions. An example of this is shown later.
- A list of target terms in the form: $2S+1\ L\ \Pi$. There must be NAST terms in this list.
- The number of base configurations from which all $(N+1)$-electron configurations are to be generated.
- The minimum occupation numbers for all $(N+1)$-electron configurations.

- The maximum occupation numbers for all $(N+1)$-electron configurations.
- A list of base configurations from which all $(N+1)$-electron configurations can be specified. The last number in this list is the maximum number of promotions from the base configuration to be allowed within the constraints specified by the minimum and maximum occupation numbers. For this simple case, the code will generate $1s^2$ plus all possible configurations resulting from the promotion of one or two electrons out of $1s^2$. For more complex cases, it is safer to input all $(N+1)$-electron configurations explicitly, each one with zero promotions. An example of this is shown later.

The $(N+1)$-electron configurations are required to compensate for the enforced orthogonality between the continuum orbitals and the bound orbitals. You should include those $(N+1)$-electron configurations that arise from adding any of the target orbitals (spectroscopic or pseudo) to any N-electron configuration. However, if your list of N-electron terms does not include all those that are possible from the specified N-electron configurations, this procedure will lead to the inclusion pseudo-resonances. The programs pstg2r.f and pstg3r.f include options designed to eliminate these pseudo resonances, but they will not be discussed here.

One might think that there are 56 partial waves for this run (4×14); however, the two partial waves ^1S and ^3S with odd parity are not possible, leaving only 54. Thus, we ran pstg2r.f on 18 processors on the Opteron, so that the calculations for three partial waves would be carried out on each processor. The command for this run is:

```
nohup time mpirun -np 18 pstg2r.x &
```

6.3. pstg3r.f

Along with the STG2HXXX.DAT binary passing files, pstg2r.f generates the formatted file sizeH.dat. For each $SL\Pi$ partial wave, this file contains three numbers: the number of free scattering channels, the size of the $(N+1)$-electron matrix without those elements arising from the $(N+1)$-electron bound terms, and the total size of the $(N+1)$-electron matrix. It is used to allocate the minimum amount of space for each Hamiltonian matrix. Other than these passing files, the only required input deck for **pstg3r.f** is **dstg3**. The **dstg3** file for this run is shown on the next page.

```
S.S. 57-term RMPS calculation of H ion
&STG3A /
&STG3B NAST=57 /
&matrixdat nb=32 nprow=5 npcol=5 /
0.000000 0.750000 0.888889 0.937500 0.960573 0.981468 1.018174 1.083642 1.214352
1.505481 2.428459 8.041553 0.750001 0.888890 0.937501 0.960654 0.981962 1.019091
1.083473 1.207405 1.463618 2.165713 5.151632 0.888891 0.937501 0.960688 0.982028
1.018493 1.079881 1.192569 1.411896 1.946720 3.811242 0.937501 0.960733 0.982331
1.018776 1.079221 1.185666 1.384806 1.829872 3.190563 0.960740 0.982465 1.019866
1.082874 1.190142 1.385829 1.795323 2.932692 0.982475 1.026724 1.103171 1.231934
1.460278 1.916327 3.088840
```

S.S. is the only required input on the first line of this file – all the rest is used for identification only.

- NAST = 0 (default) causes no energy adjustment of the theoretical term energies. If NAST $>$ 0, then the code expects the user to input energies in rydbergs following the last namelist input for each of the terms specified in dstg2 and in the order in which they were read in dstg2, unless ISORT=1 (as in this hydrogen example) in which case they are grouped together by symmetry and each symmetry is in the same order in which it first appeared in the dstg2 input. The code adjusts the diagonal elements of the continuum-continuum part of the $(N + 1)$-electron Hamiltonian by the differences between the input energies and the theoretical energies. This is normally used to adjust the theoretical values to the experimental values; however, here it is being used to remove the degeneracy between energies in the spectroscopic terms in hydrogen, since these can cause errors in the calculation.
- NB = Global column block size for partitioning the global Hamiltonian matrix. When the majority of matrices in the calculation are less than size 10,000, then a value of 16 is advisable. When the largest matrices range in size between 10,000 and 35,000 a value of 32 should be used, and for even larger cases it might be set equal to 64. However, a value of 32 has been found to work with a wide variety of matrix sizes.
- NPROW & NPCOL = variables used to determine the dimensions of the local Hamiltonian matrix distributed to each processor. **NPROW*NPCOL must equal the total number of processors** and the code runs more efficiently when NPROW = NPCOL.

The run command for pstg3r.f on 25 Opteron processors is as follows:

```
nohup time mpirun -np 25 pstg3r.x &
```

This concludes the inner-region exchange calculation. The input files and the NX1.DAT, NX2.DAT, and H.DAT output files should be saved. To save disk space, all other passing files can be deleted. The files NX1.DAT, NX2.DAT and dstg3 files need to copied into a separate directory in which the non-exchange calculation is to be carried out.

6.4. the non-exchange run: pstg1nx.f, pstg2nx.f and pstg3nx.f

The flow chart for a non-exchange calculation is shown in figure 2. With the exception of dstg3, which is needed for pstg3nx.f, and the two passing files NX1.DAT and NX2.DAT from the exchange runs, the only additional input file needed for the non-exchange runs is **dstgnx**. The **dstgnx** file for this case is shown below:

```
CONTinuation of RMPS calculation of H
&STGNX MINLT=14 MAXLT=60 /
```

The only part of the first line of this file that is required is the first four letters CONT, indicating this non-exchange run is a continuation of the exchange run. The rest of the line is used for identifying the run.

- MINLT= minimum angular momentum for the non-exchange calculation. For LS-coupling calculations, it is invariably MAXLT from the exchange calculation plus one.
- MAXLT = maximum angular momentum for the non-exchange calculation.

The simplicity of the non-exchange input results from the fact that **dstgnx** is used for pstg1nx.f , pstg2nx.f and pstg3nx.f. These codes are run in 1, 2, 3 order. Be sure to keep the H.DAT file resulting from the non-exchange run in a directory separate from the H.DAT resulting from the exchange run.

6.5. Outer region run: pstgf.f

In general one would want to do four pstgf.f runs – two for the exchange calculation and two for the non-exchange calculation. The first run for the exchange calculation would attempt to resolve the resonance structure by employing a very fine energy mesh over the energy region of the spectroscopic thresholds. The second run for the

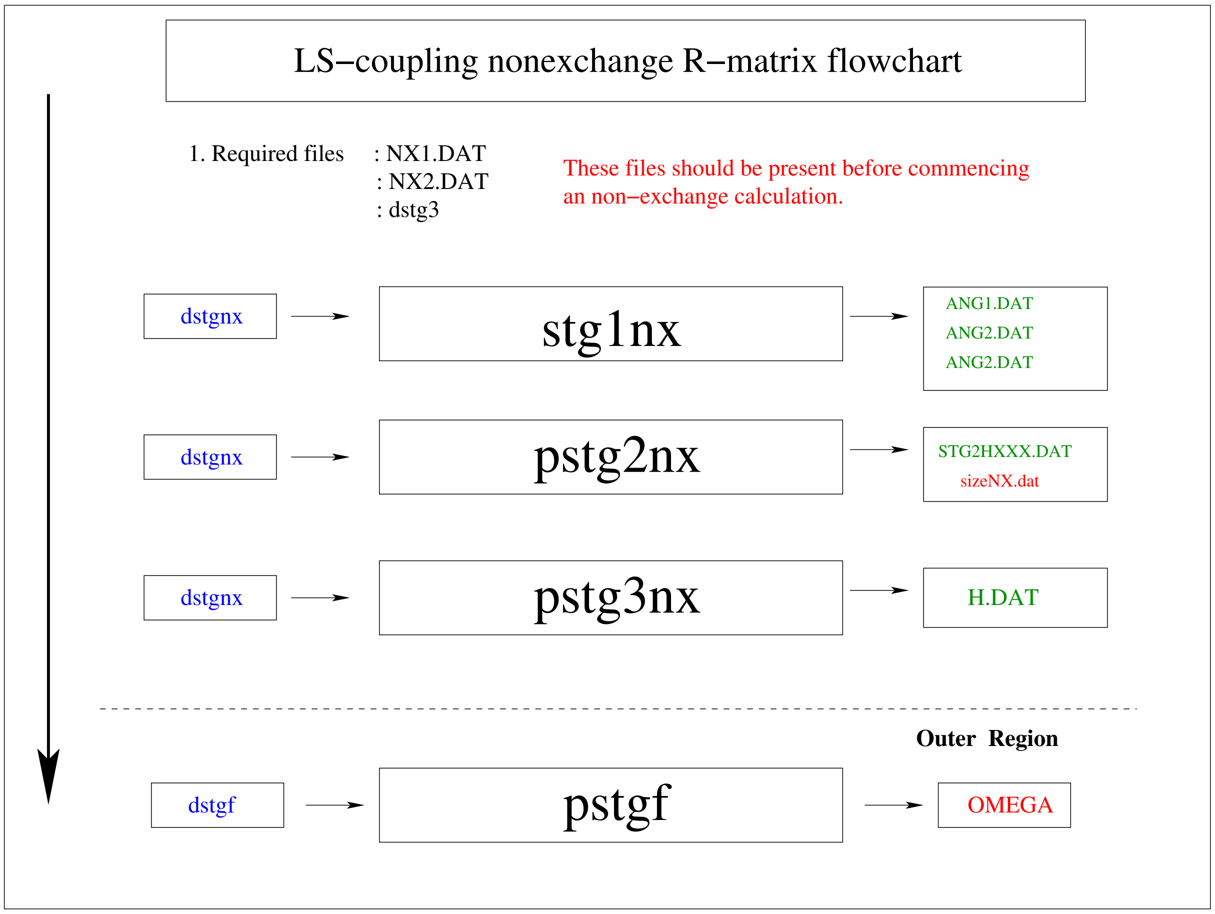

LS−coupling nonexchange R−matrix flowchart
1. Required files : NX1.DAT
: NX2.DAT
: dstg3
These files should be present before commencing an non−exchange calculation.
dstgnx
stg1nx
ANG1.DAT
ANG2.DAT
ANG2.DAT
dstgnx
pstg2nx
STG2HXXX.DAT
sizeNX.dat
dstgnx
pstg3nx
H.DAT
Outer Region
dstgf
pstgf
OMEGA

Figure 2. Flowchart of the LS-coupling non-exchange R-matrix inner region codes. The output files in green are binary and the ones in red are formatted.

exchange calculation would employ a much coarser mesh over the energy range above the highest spectroscopic threshold. Both of the non-exchange runs would normally use a relatively coarse energy mesh; however, it is still useful to divide these runs in two in order that the lower energy one is sufficiently fine to pick up the spectroscopic thresholds. Other than the H.DAT file, the only input file needed is **dstgf**. The **dstgf** file for H in the low-energy resonance region that was used in conjunction with the H.DAT file from the exchange calculation is shown on the next page.

```
&STGF IMESH=1 IQDT=0 PERT='YES' MAXLT=13 LRGLAM=-13 IPRINT=-2
IBIGE=0 /
&MESH1 MXE=1000 E0=0.7499 EINCR=2.5035e-04 /
```

- IQDT controls the nature of the outer region calculation.
  IQDT = 0 is the default and is used to specify the standard pstgf.f operation.
  IQDT = 1 specifies the Multi-channel Quantum Defect Theory (MQDT) operation via the use of unphysical S-matrices.
  IQDT = 2 specifies the MQDT operation via the use of unphysical K-matrices. All examples shown in the present document use only the default option.
- IMESH defines the type of energy mesh and subsequent namelist to be read.
  IMESH = 1, constant spacing in energy dE.
  IMESH = 2, constant spacing in effective quantum number dn.
  IMESH = 3, an arbitrary set of user-supplied energies.
- PERT Determines if the long-range multipoles are included perturbatively in the solution to the scattering problem in the asymptotic region.
  PERT = 'NO', neglect long-range coupling potentials (fast operation).
  PERT = 'YES', include long-range coupling potentials (factor 5-10 slower). Even though it requires much more time, we would recommend that PERT always be set to YES. Neglecting these long-range multipoles can have a large effect on the collision strengths, even for high partial waves.
- LRGLAM Determines if top up is to be included in the collision strengths.
  LRGLAM >= 0 is the maximum L-value for an *LS* calculation, or maximum value of 2*J for a BP calculation – partial wave sum will be topped-up in order to estimate the contributions to the collision strengths for partial waves from LRGLAM+1 to infinity.
  LRGLAM < 0 (default) No top-up.
  In the hydrogen case, we would set LRGLAM negative for the exchange runs, but equal to 60 for the non-exchange runs, so that the high partial-wave contributions would include an estimate for the partial waves with $L > 60$.
- IBIGE determines if infinite-energy limits are determined for dipole-allowed transitions.
  IBIGE = 0, default, no limits are calculated
  IBIGE = 1, append Bethe infinite-energy scaled collision strengths to the file OMEGA. (There is now an option in AUTOSTRUCTURE that allows for the generation of infinite-energy scaled collision strengths for dipole and non-dipole transitions, and this is normally used instead of generating only the dipole-allowed

values from pstgf.f)

- IPRINT defines the print level, -2 is lowest level and +3 the highest.
- IOPT1 Option to allow for a reduced number of partial waves.
  IOPT1 = 1 is the default, and all $SL\Pi$ partial waves are included.
  IOPT1 = 2 specify partial waves in the form, $2S + 1\ \ L\ \ \Pi$ (or $0\ \ 2J\ \ \Pi$ for a BP calculation) after the last namelist input. The list of partial waves must be terminated with -1 -1 -1.
- MXE = the number of energy mesh points.
- E0 = the first energy point in scaled energy units.
- EINCR = the energy increment in scaled energy units.

Note that by scaled energy units, we mean the energy of the incident electron relative to the ground state in rydbergs/(Z-N)^2. The code pstgf.f is used for both exchange and non-exchange calculations. However, as mentioned previously, the non-exchange calculation should have substantially fewer energy points. It is also very important to note that pstgf.f requires that the **total number of energy points should be divisible by the number of processors**. The run command for pstgf.f on 25 Opteron processors is shown below

```
nohup time mpirun -np 25 pstgf.x &
```

Once a pstgf.f calculation has been completed, numerous OMEGAXXX files will be present within the directory. Running the utility program arrange.f (no input file needed) will generate a single OMEGAZ file, containing energy-ordered collision strengths for every possible transition.

6.6. The utility code: omadd.f

The utility code omadd.f may now be used to remove isolated numerical failures, run tests for unresolved resonances, and add collision strengths for the high partial waves from the non-exchange run to the collision strengths from the lower partial waves determined from the exchange run. A sample input file (**dadd**) for omadd.f used to remove isolated numerical failures in the OMEGA file that was generated from the exchange run in the resonance region is shown below:

```
&SADD YMULT=1000.0 /
```

One first copies the OMEGA file to be processed into omadd1. After omadd.f is run, the processed collision strengths appear in omaddt. The YMULT option causes the code to search through the data in the OMEGA file and examine the collision strengths in groups at three consecutive energy points; should the collision strength for the middle point differ in magnitude from collision strengths at the points on either side by a factor of YMULT or more, then the middle point is removed and replaced by a collision strength determined from the average of the collision strengths on either side. It is unclear exactly what value of YMULT one should use to eliminate numerical failures without removing important physics, but a value of 1000 appears fairly safe in the resonance region. For an OMEGA file containing collision strengths in the high energy region above all thresholds, or for an OMEGA file with only collision strengths for high partial waves, there are no physical resonances; in such cases, a value much closer to one such as 1.2 is appropriate.

One can also employ the YMULT option to test for unresolved resonances. For example, one could copy the OMEGA file run in the resonance region after it has been processed with a YMULT value of 1000 into omadd1 and then run omadd.f with a YMULT value of 5 or 10. By comparing the collision strengths from the two files one can arrive at some conclusions regarding the effects from unresolved resonances and perhaps decide whether or not a finer energy mesh is needed.

After filtering the OMEGA files in the low and high energy regions from both the exchange and non-exchange calculations, the next step is to merge these files. There are a number of ways to do this, but perhaps the easiest way is to copy the low-energy OMEGA file into OMEGA000 and the high energy one into OMEGA001 and then run arrange.f $--$ OMEGAZ will then contain the merged files.

omadd.f can finally be used to add the contributions from the lower partial waves from the exchange run with those for the higher partial waves from the non-exchange run. This is done by copying the OMEGA file from the exchange run into omadd1 and the OMEGA file from the non-exchange run into omadd2. For this case, the dadd file should not include a value for YMULT. The code assumes that the data in omadd2 are on a coarser energy mesh than those in omadd1. It then interpolates the collision strengths from omadd2 onto the mesh used in omadd1 and adds the collision strengths. The output data on the energy mesh employed in omadd1 is written into omaddt.

6.7. The utility code: stgsig.f

Once the final file has been generated, one would normally want to determine the collision strengths or cross sections as a function of incident electron energy. The utility code stgsig.f is designed to do this. It also has a number of options for doing such things as adding cross sections for various transitions and generating reduced cross sections as a function of reduced energy, but those will not be discussed here.

Suppose,we wish to generate the excitation cross sections for all transitions from 1s → 2s to 1s → 3d in Mb (10^{-18} cm^2) as a function of energy in eV over an energy range from 0 to 50 eV. The **dstgsig** input file for such a run with stgsig.f is shown below:

```
&stgsig ntran=-5 ntrmn=1 units=13.606 isgunt=0 emn=0. emx=50. ic=0 /
```

- ntran - final transition for which collision strengths or cross sections are to be output. It should be emphasized that this denotes the final transition number and not the final term. The minus sign indicates that cross sections will be displayed, whereas a postive number would return collision strengths.
- nast - the initial transition for which collision strengths or cross sections are to be output.
- units - energy units: 1.0 for Rydbergs and 13.606 for eV (default).
- isgunt - units for cross sections: isgunt = 0 for 10^{-18} cm^2 (default); isgunt = 1 for 10^{-16} cm^2; isgunt = 2 for πa_0^2; isgunt = 3 for a_0^2; and isgunt = -1 for 10^{-21} cm^2.
- emn - minimum energy of the incident electron in units.
- emx - maximum energy of the incident electron in units.
- ic - coupling; 0 for an *LS* coupling OMEGA file (default) ; 1 for an intermediate coupling BP or DARC OMEGA file.

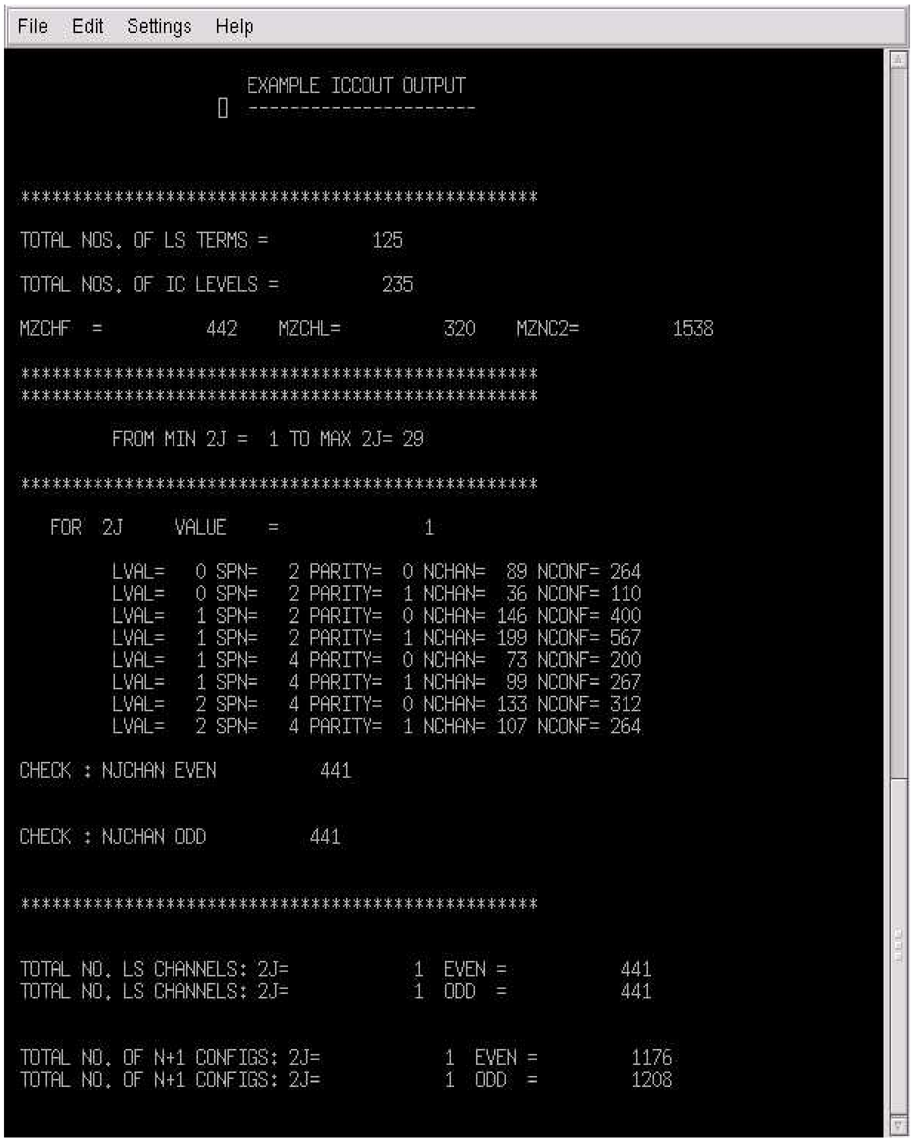
The output is written to the file sg.dat.

7. Breit-Pauli calculations

7.1. Introduction

We suggest that you consider the following steps in running a BP calculation. The first step is necessary only when the dimensions set in the PARAM file must be very accurate to allow the executables to run on machines with a limited amount of memory for each processor.

- Run the first two stages pstg1r.f and pstg2r.f on one processor. In the dstg1 file, the MAXC parameter determining the span of your continuum basis should be set very low; e,g., 4-5. Then, within the STG2B namelist in the dstg2 input file set set NOICC = 1 (Number Of Intermediate Coupling Channels); this will produce an



```
                EXAMPLE ICCOUT OUTPUT
                ---------------------


**************************************************

TOTAL NOS. OF LS TERMS =         125

TOTAL NOS. OF IC LEVELS =         235

MZCHF  =         442    MZCHL=         320    MZNC2=        1538

**************************************************
**************************************************

         FROM MIN 2J =  1 TO MAX 2J= 29

**************************************************

   FOR  2J     VALUE    =             1

         LVAL=   0 SPN=   2 PARITY=  0 NCHAN=  89 NCONF= 264
         LVAL=   0 SPN=   2 PARITY=  1 NCHAN=  36 NCONF= 110
         LVAL=   1 SPN=   2 PARITY=  0 NCHAN= 146 NCONF= 400
         LVAL=   1 SPN=   2 PARITY=  1 NCHAN= 199 NCONF= 567
         LVAL=   1 SPN=   4 PARITY=  0 NCHAN=  73 NCONF= 200
         LVAL=   1 SPN=   4 PARITY=  1 NCHAN=  99 NCONF= 267
         LVAL=   2 SPN=   4 PARITY=  0 NCHAN= 133 NCONF= 312
         LVAL=   2 SPN=   4 PARITY=  1 NCHAN= 107 NCONF= 264

CHECK : NJCHAN EVEN          441


CHECK : NJCHAN ODD          441


**************************************************


TOTAL NO. LS CHANNELS: 2J=           1  EVEN =          441
TOTAL NO. LS CHANNELS: 2J=           1  ODD  =          441


TOTAL NO. OF N+1 CONFIGS: 2J=           1  EVEN =        1176
TOTAL NO. OF N+1 CONFIGS: 2J=           1  ODD  =        1208
```

Figure 3. The file ICCOUT used to check the DIMENSIONS in precupd.f

output file ICCOUT, which gives the values of the minimum dimensions required for the BP run. (A sample ICCOUT file is shown in figure 3.) This allows one who is using a machine with limited memory per processor to edit the PARAM file accordingly and create the smallest possible executable.

- Having specified the values in the PARAM file (perhaps using the information in the ICCOUT file) and compiling the inner region LS/BP codes, return to the dstg1 file, and set the MAXC parameter to a realistic value. The parameter RELOP must be set to RELOP = 'YES' in order for pstg1r.f to calculate the mass-velocity, Darwin, and spin-orbit integrals needed in a BP calculation.

- On completion of pstg1r.f there should be a number of RKXXX.DAT files of nearly equal size. The next step is to run pstg2r.f. In the dstg2 file, be sure that the NOICC flag is now removed and the RELOP parameter is set to RELOP='YES' within the STG2A namelist. Ideally, it is best to run a single $SL\Pi$ partial wave per processor, but if this is not possible, try to divide the number of partial waves as evenly as possible among the available processors.
- Next you will run pstg2.5.f to create a set of files containing information on the LS $(N+1)$-electron matrix elements that are required for a particular range of $J\Pi$ partial waves. There are two input variables for pstg2.5.f and precupd.f that need explaining. The variables J2MIN and J2MAX specify the minimum and maximum values of $2 \times J$ and the total number of partial waves will be (J2MAX-J2MIN+2). Ideally, one should run one $J\Pi$ partial wave per processor; however, if this is not feasible, **the total number of partial waves must be exactly divisible by the number of processors. Also note that both pstg2.5.f and precupd.f must be run with the same number of processors.**
- both pstg2.5.f and precupd.f read the input deck **dstgjk**. After you have completed running pstg2.5.f, you will have a substantial number of large passing files. The RKXXX.DAT files produced by pstg1r.f and the STG2HXXX.DAT files produced by pstg2r.f can now be removed.
- After you have completed precupd.f you can remove the **STG2HJXXX** files. All the $J\Pi$ partial waves will be listed in ascending order within the sizeBP.dat file. Although one would normally want to run pstg3r.f to diagnonalize the $(N+1)$ matrices for all these partial waves, under certain circumstances you may wish to run pstg3r.f for a subset of partial waves. This can be done by creating a new sizeBP.dat with the partial waves to be run and renaming the **RECUPHXXX** files appropriately.

A flow diagram for a parallel BP R-matrix run is shown in figure 4. In this diagram, it is assumed that the target structure has been generated using the program AUTOSTRUCTURE.

7.2. A Breit-Pauli example : Fe^{14+}

We will now consider a BP calculation on Fe^{14+} using the IBM SP computer at NERSC; it uses the batch processing queue system Loadleveler to manage jobs. This

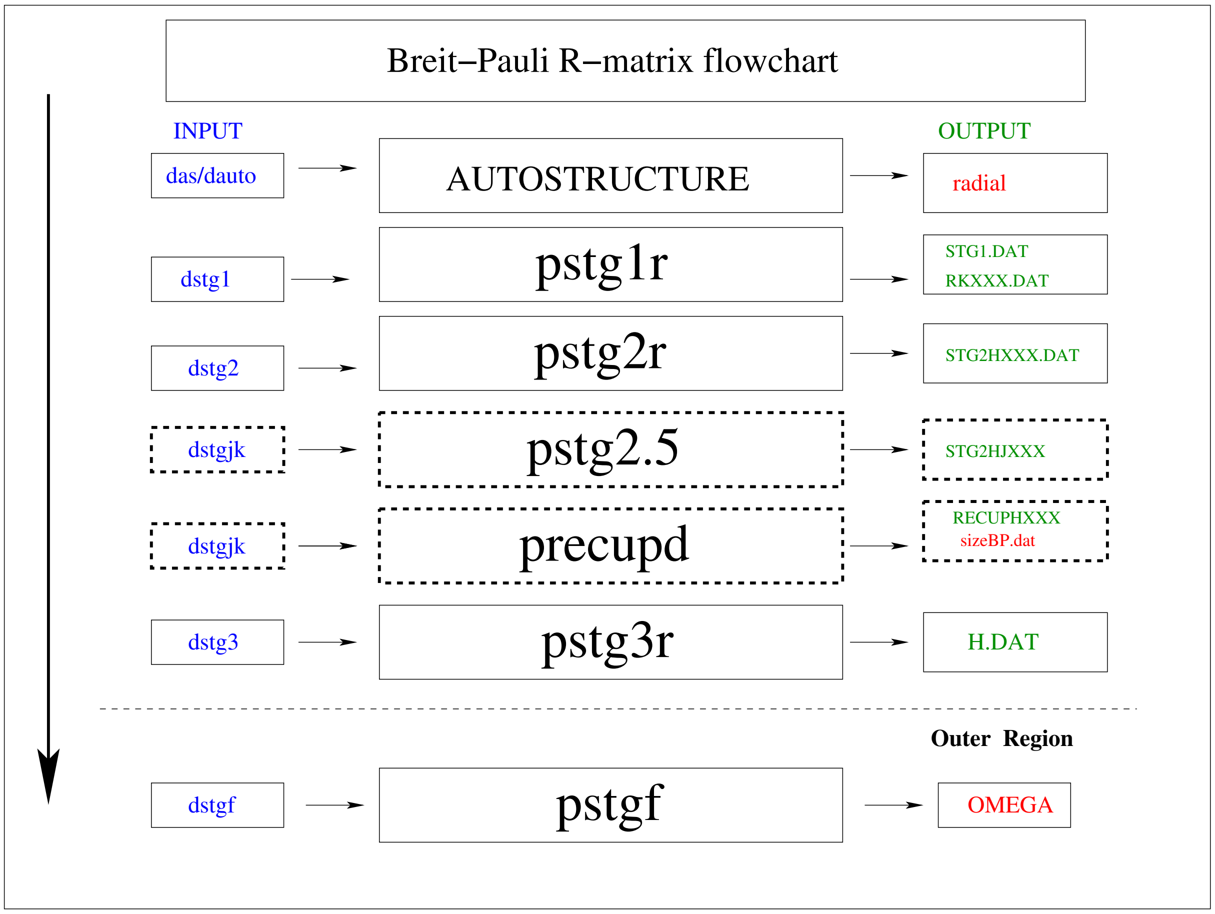


Figure 4. Flowchart of parallel R-matrix inner-region codes; the green output files are binary and red ones are formatted files

will be demonstrated with some sample batch scripts. The **dstg1** input file for the pstg1r.f run is shown below. With the exception of the RELOP command that signifies a BP run, it is quite similar to the dstg1 file shown in the last section.

```
S.S. 25 term, 45-level BP calculation for Fe14+
&STG1A RELOP='YES' /
&STG1B MAXLA=4 MAXLT=15 MAXE=300 MAXC=25 /
```

In this case, we wish to generate all $J\Pi$ partial waves up to a $J = 13.5$. This requires us to generate all $SL\Pi$ partial waves up to an $L = 15$; thus, MAXLT $= 15$.

A batch script for running pstg1r.f on the IBM-SP is shown below. These batch jobs are submitted to the available queues with the command '**llsubmit** *scriptfile-name*', where the *class* and *wall_clock_limit* parameters determine the queue in which the job will be run. We made the pstg1r.f run on one node of the IBM-SP using 16 processors. Details regarding the Loadleveler commands are given on the NERSC Web site.

```
#!/usr/bin/csh
#@ job_name = pstg1r
#@ output = pstg1r.out
#@ error = pstg1r.error
#@ job_type = parallel
#@ network.MPI = csss,not_shared,us
#@ notification = never
#@ class = regular
#@ node = 1 !! 16 processors
#@ total_tasks = 16
#@ wall_clock_limit = 00:30:00 !! 30 minutes
#@ queue
setenv TMPDIR $SCRATCH
cd /scratch/scratchdirs/cball/fe14+bp
pwd
poe ./pstg1r.x -procs 16 -nodes 1
```

The commands '**llq -u** *username*' and '**llcancel** *job id*' are useful to monitor or cancel jobs in the batch queue.

The input file **dstg2** for the pstg2r.f run for Fe^{14+} is given below:

```
S.S. 25 term, 45-level BP calculation for Fe14+
&STG2A RELOP='YES' /
&STG2B MAXORB=9 NELC=12 NAST=25 INAST=0 MINLT=0 MAXLT=15
MINST=2 MAXST=4 /
1 0 2 0 2 1 3 0 3 1 3 2 4 0 4 1 4 2        ! nl orbitals
9                 ! 9 N-electron configurations explicitly written out
2 2 6 0 0 0 0 0 0           ! min electron occupation of a shell
2 2 6 2 2 2 1 1 1           ! max electron occupation of a shell
2 2 6 2 0 0 0 0 0 0
2 2 6 1 1 0 0 0 0 0
```

```
2 2 6 0 2 0 0 0 0 0
2 2 6 1 0 1 0 0 0 0
2 2 6 0 0 2 0 0 0 0
2 2 6 1 0 0 1 0 0 0
2 2 6 1 0 0 0 1 0 0
2 2 6 1 0 0 0 0 1 0
2 2 6 0 1 1 0 0 0 0
1 0 0
1 0 0
1 0 0
1 0 0
3 1 1
3 1 1
3 1 1
1 1 1
1 1 1
1 1 1
1 2 0
1 2 0
1 2 0
1 2 0
3 1 0
3 1 0
3 2 0
3 2 0
3 3 1
1 2 1
3 2 1
1 3 1
3 3 0
1 4 0
3 0 0
33                  ! 33 (N+1)-electron configurations explicitly written out
2 2 6 0 0 0 0 0 0           ! min electron occupation of a shell
2 2 6 2 3 3 2 2 2           ! max electron occupation of a shell
2 2 6 2 1 0 0 0 0 0
2 2 6 2 0 1 0 0 0 0
2 2 6 2 0 0 1 0 0 0
```

```
2 2 6 2 0 0 0 1 0 0
2 2 6 2 0 0 0 0 1 0
2 2 6 1 2 0 0 0 0 0
2 2 6 1 1 1 0 0 0 0
2 2 6 1 1 0 1 0 0 0
2 2 6 1 1 0 0 1 0 0
2 2 6 1 1 0 0 0 1 0
2 2 6 0 3 0 0 0 0 0
2 2 6 0 2 1 0 0 0 0
2 2 6 0 2 0 1 0 0 0
2 2 6 0 2 0 0 1 0 0
2 2 6 0 2 0 0 0 1 0
2 2 6 1 0 2 0 0 0 0
2 2 6 1 0 1 1 0 0 0
2 2 6 1 0 1 0 1 0 0
2 2 6 1 0 1 0 0 1 0
2 2 6 0 1 2 0 0 0 0
2 2 6 0 1 1 1 0 0 0
2 2 6 0 1 1 0 1 0 0
2 2 6 0 1 1 0 0 1 0
2 2 6 0 0 3 0 0 0 0
2 2 6 0 0 2 1 0 0 0
2 2 6 0 0 2 0 1 0 0
2 2 6 0 0 2 0 0 1 0
2 2 6 1 0 0 2 0 0 0
2 2 6 1 0 0 1 1 0 0
2 2 6 1 0 0 1 0 1 0
2 2 6 1 0 0 0 2 0 0
2 2 6 1 0 0 0 1 1 0
2 2 6 1 0 0 0 0 2 0
```

Note that unlike our RMPS calculation for H, we are now specifying the N-electron and $(N+1)$-electron configurations explicitly, and therefore, the number of promotions for each configuration is set to zero. Since there are 64 $SL\Pi$ partial waves, we ran pstg2r.f on 64 processors. The batch script for this run is shown on the next page.

```
#!/usr/bin/csh
#@ job_name = pstg2r
#@ output = pstg2r.out
#@ error = pstg2r.error
#@ job_type = parallel
#@ network.MPI = csss,not_shared,us
#@ notification = never
#@ class = regular
#@ node = 4 !! 64 processors
#@ total_tasks = 64
#@ wall_clock_limit = 00:30:00 !! 30 minutes
#@ queue
setenv TMPDIR $SCRATCH
cd /scratch/scratchdirs/cball/fe14+bp
pwd
poe ./pstg2r.x -procs 64 -nodes 4
```

Once the pstg2r.f run has been completed and the STG2HXXX.DAT files have been generated, we are ready to run pstg2.5.f to organize the $SL\Pi$ partial-wave information in the files STG2HJXXX files for a precupd.f run. As mentioned above, pstg2.5.f and precupd.f both use the input file **dstgjk**. The **dstgjk** file for this Fe^{14+} run is shown below. Since we wish to generate $J\Pi$ partial waves from $J = 0.5$ to $J = 13.5$, J2MIN is set to 1 and J2MAX is set to 27. The total number of partial waves will be 28. Thus we made our pstg2.5.f run using 28 processors.

```
S.S. 25 term, 45-level BP calculation for Fe14+
&STGJA RELOP='YES' /
&STGJB JNAST=45 IJNAST=0 J2MIN=1 J2MAX=27 /
0 0
0 1
2 1
4 1
2 1
0 0
4 0
4 0
```

```
0 0
2 0
4 0
6 0
4 0
4 1
6 1
4 1
8 1
2 1
4 1
6 1
0 1
2 1
4 1
6 1
2 1
4 0
6 0
8 0
4 0
0 0
4 0
2 0
8 0
0 0
2 0
0 0
0 1
2 1
4 1
2 1
2 0
4 0
6 0
4 0
```

In this input file:

- JNAST = number of target levels, specified as $2 \times J\ \Pi$ after the namelist.
- IJNAST = number of $J\Pi$ partial waves. By setting IJNAST = 0, the code will generate all possible partial waves from J2MIN to J2MAX.
- J2MIN = minimum total value of $2 \times J$ (provided IJNAST=0)
- J2MAX = maximum total value of $2 \times J$ (provided IJNAST=0)

The batch script for the pstg2.5.f run on 28 processors (2 nodes) on the IBM-SP is shown below:

```
#!/usr/bin/csh
#@ job_name = pstg2.5
#@ output = pstg2.5.out
#@ error = pstg2.5.error
#@ job_type = parallel
#@ network.MPI = csss,not_shared,us
#@ notification = never
#@ class = regular
#@ node = 2 !! 28 processors
#@ total_tasks = 28
#@ wall_clock_limit = 00:30:00 !! 30 minutes
#@ queue
setenv TMPDIR $SCRATCH
cd /scratch/scratchdirs/cball/fe14+bp
pwd
poe ./pstg2.5.x -procs 28 -nodes 2
```

Once pstg2.5.f has been run and the STG2HJXXX files have been generated, precupd.f is run using the identical dstgjk input file. The batch script for the precupd.f run is the same as the one above, except the code name pstg2.5.f would be changed to precupd.f. Remember that precupd.f must be run on the same number of processors as pstg2.5.f

The program precupd.f produces a formatted file sizeBP.dat. For each $J\Pi$ partial wave, it lists three quantities: the number of free scattering channels, the size of the $(N+1)$-electron Hamiltonian matrix without those elements arising from the $(N+1)$-electron bound levels, and the total size of the $(N+1)$-electron Hamiltonian matrix. Along with the binary RECUPHXXX files, this file must be present, when pstg3r.f is run; it is used to allocate the minimum amount of space for each Hamiltonian matrix.

The **dstg3** input deck for the pstg3r.f run is shown below:

```
S.S. 25 term, 45-level BP calculation for Fe14+
&STG3A /
&STG3B INAST=0 NAST=0 /
&MATRIXDAT NB=32 NPROW=8 NPCOL=8 /
```

In this run, we are not making any adjustments to the theoretical energies. However, if such adjustments were desired, one would set NAST to the number of levels included in the close-coupling expansion of the N-electron target and the adjusted energies would be entered following the last namelist input line. The order of these energies would now be determined by the order of the levels in the dstgjk input file.

Since we have set NPROW×NPCOL = 64, pstg3r.f must be run on 64 processors. The batch script for this run is shown below:

```
#!/usr/bin/csh
#@ job_name = pstg3r
#@ output = pstg3r.out
#@ error = pstg3r.error
#@ job_type = parallel
#@ network.MPI = csss,not_shared,us
#@ notification = never
#@ class = regular
#@ node = 4 !! 64 processors
#@ total_tasks = 64
#@ wall_clock_limit = 01:00:00 !! 1 hour
#@ queue
setenv TMPDIR $SCRATCH
cd /scratch/scratchdirs/cball/fe14+bp
pwd
poe ./pstg3r.x -procs 64 -nodes 4
```

There are no differences in the **dstgf** input file for this BP pstgf.f run than the one shown for the LS-coupling H run in the last section, with the exception that MAXLT and LRGLAM now refer to $2 \times J$ rather than L.

8. Dirac Atomic R-Matrix Code Calculations

8.1. Introduction

As mentioned above, our version of the inner-region Dirac Atomic R-matrix Code (DARC) allows us to employ **pstg3r.f** and **pstgf.f** to carry out the diagonalization of the $(N + 1)$-electron matrix and solve the scattering problem in the asymptotic region, respectively. This required the development of a new code pdto3.f and some modifications of pstg3r.f and pstgf.f to handle the quantum numbers that describe each channel for a relativistic calculation. As with an LS or Breit-Pauli R-matrix calculation, the accuracy of a DARC calculation depends significantly on the quality of the N-electron target states. The DARC codes assume that the target orbitals have been generated using the GRASP code of Ian Grant *et al* . The version of GRASP that we have employed for this purpose is GRASP^0 and it is available along with the original serial version of DARC at *http : //web.am.qub.ac.uk/DARC/*

8.2. A DARC example: Fe^{14+}

It is not our purpose here to give complete instructions for the use of GRASP and DARC, since they are available at the Web address given above; rather, here we will only provide the input files used to make a DARC run for Fe^{14+} using our parallel version of the DARC package. As we go along, we will highlight the parallel features that we have implemented.

The input file **GRASP.INP** associated with with the program grasp0.f to generate the relativistic target states for a 45-level DARC calculation for Fe^{14+} is shown below. The code allows one to specify the non-relativistic orbitals (in this case, 1s, 2s, 2p, 3s, 3p, 3d, 4s, 4p, and 4d) and the non-relativistic configurations (in this case, 3s^2, 3s3p, 3s3d, 3p^2, 3p3d, 4s^2, 4s4p, and 4s4d) to internally generate all the corresponding relativistic orbitals and configurations.

```
Orbitals for a 45-level DARC calculation on Fe14+
9 9          ! NMAN, NWM
1S 2 2 2 2 2 2 2 2 2
2S 2 2 2 2 2 2 2 2 2
2P 6 6 6 6 6 6 6 6 6
3S 2 1 1 0 0 0 1 1 1
3P 0 1 0 2 1 0 0 0 0
3D 0 0 1 0 1 2 0 0 0
4S 0 0 0 0 0 0 1 0 0
```

```
4P 0 0 0 0 0 0 0 1 0
4D 0 0 0 0 0 0 0 0 1
ANG 7 10
-1                  ! all poss J Π
MCP
MCT 1 -1 -1             ! E1 only
MCBP 0
MCDF
0 0 0           ! initial orbitals are generated internally
26
EAL 3
BENA 16

OSCL 9 10 11

STOP
```

The GRASP code has an extremely wide range of options, and before running this program, one should carefully study the GRASP manual. This minimal input file is only provided for illustrative purposes.

After GRASP has been run and the target orbitals written to the file ORBITALS.DAT, one must run the interactive program **stg1d0.f** to generate the file TARGET.INP, which contains the Dirac two-component (large and small) radial wavefunctions in a format that can be read by DARC. It is a stand alone code with no input deck required, and the user need only input the variant of GRASP that is being used. In this case we entered 1, since that corresponds to the grasp0.f version that we are using.

In figure 5 is shown a flow diagram for a typical GRASP/DARC run using our suite of parallel DARC codes as well as **pstg3r.f** and **pstgf.f**.

We first note that unlike the LS/BP R-matrix program pstg1r.f, the tasks of generating the continuum orbitals and calculating the radial integrals have been separated into two programs within DARC. After the orbital file TARGET.INP has been generated, we are ready to run the program **stg1d_orb.f** to calculate the $(N+1)$ continuum orbitals. It reads the input deck ORBS.INP, and the orbital file TARGET.INP. The file **ORBS.INP** that we employed for our Fe^{14+} run is shown on the next page. Here we use the same number of basis functions to represent the $(N+1)$-electron continuum as we employed in our BP calculation for Fe^{14+} (NRANG2 = 25).

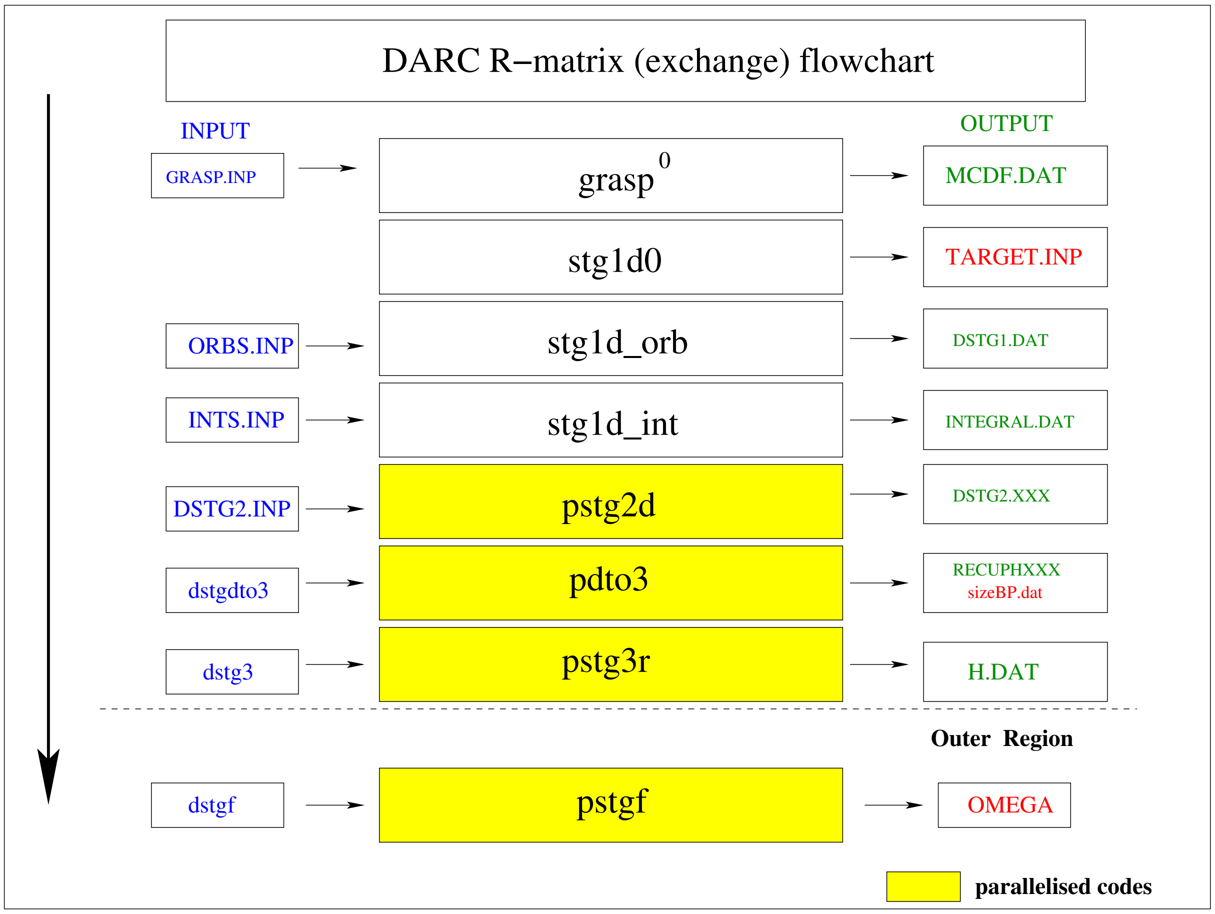


Figure 5. Flowchart of the GRASP and the DARC R-matrix inner region codes. The codes that have been parallelized, including pstg3r.f and pstgf.f are shown in yellow. The binary output files are in green and the formatted output files in red.

---

title : 45-level DARC calculation for Fe 14+
&ORBS MINNQN = 1 2 2 3 3 MAXNQN = 4 4 4 4 4 MAXFUL = 2 2 2 2 2
KBMAX=5 KCMAX=37 NRANG2=25 NZ=26 NELC=12 /

---

stg1d.f runs for all but the largest cases in about a minute.

Next we run **stg1d_int.f** to calculate all one and two electron integrals and then write them to the binary file INTEGRAL.DAT. Considering the substantial increase in the number of integrals over the LS/BP case this code runs very efficiently and has not yet been parallelized. Only a minimal amount of information is required in the input deck **INTS.INP** as illustrated below:

```
title : 45-level DARC calculation for Fe 14+
&INTS LAMBB = 2 /
```

The next step is to run the code **pstg2d.f** in order to calculate the angular coefficients and form the $(N+1)$-electron matrix elements. This code has been parallelised over the $J\Pi$ partial waves. Ideally, the input file **dstgdto3** for the code **pdto3.f** is present (a sample of this is shown below where we discuss running pdto3.f); if so, it informs pstg2d.f of the number of $J\Pi$ partial waves used in advance through the namelist variable INAST. Otherwise, pstg2d.f must read the input deck **DSTG2.INP** repeatedly to ensure that each processor is positioned correctly to read only those $J\Pi$ partial waves it is calculating. Multiple DSTG2.XXX files are produced, and the $J\Pi$ partial waves are distributed as evenly as possible among the available processors. The **DSTG2.INP** input deck for this Fe^{14+} run is shown below:

```
title : 45-level DARC calculation for Fe XV
&DSTG2 NMAN=9 NWM=9 /
&ORB PRINC=1 KAPPA=-1 /
&ORB PRINC=2 KAPPA=-1 /
&ORB PRINC=2 KAPPA=-2 /
&ORB PRINC=3 KAPPA=-1 CSF= 2 1 1 1 1 1 0 0 0 /
&ORB PRINC=3 KAPPA=-2 CSF= 0 1 0 0 0 0 2 1 0 /
&ORB PRINC=3 KAPPA=-3 CSF= 0 0 1 0 0 0 0 1 2 /
&ORB PRINC=4 KAPPA=-1 CSF= 0 0 0 1 0 0 0 0 0 /
&ORB PRINC=4 KAPPA=-2 CSF= 0 0 0 0 1 0 0 0 0 /
&ORB PRINC=4 KAPPA=-3 CSF= 0 0 0 0 0 1 0 0 0 /
&ANGOPT /
&JVALUE /
&SYM JTOT=0.5 NPTY=1 /
&SYM JTOT=0.5 NPTY=-1 /
&SYM JTOT=1.5 NPTY=1 /
&SYM JTOT=1.5 NPTY=-1 /
&SYM JTOT=2.5 NPTY=1 /
&SYM JTOT=2.5 NPTY=-1 /
&SYM JTOT=3.5 NPTY=1 /
&SYM JTOT=3.5 NPTY=-1 /
&SYM JTOT=4.5 NPTY=1 /
```

```
&SYM JTOT=4.5 NPTY=-1 /
&SYM JTOT=5.5 NPTY=1 /
&SYM JTOT=5.5 NPTY=-1 /
&SYM JTOT=6.5 NPTY=1 /
&SYM JTOT=6.5 NPTY=-1 /
...skipping a few
&SYM JTOT=13.5 NPTY=1 /
&SYM JTOT=13.5 NPTY=-1 /
```

After the **pstg2d.f** run has been completed, one must run the program **pdto3.f**. This is a newly developed utility code originally written by Prof Keith Berrington. For each $J\Pi$ symmetry, it reads the $(N+1)$-electron matrix elements that have been written to the files DSTG2.XXX, and writes them to the files RECUPHXXX that can be read by pstg3r.f. The input deck **dstgdto3** for the code pdto3.f has the single variable INAST, which corresponds to the total number of symmetries listed in the input deck DSTG2.INP. In the serial version of this code, the total number of symmetries is input interactively. However, for the parallel implementation, it is better to enter this information using an input file. The number of partial waves contained within each RECUPHXXX file is equal to the number of partial waves DSTG2.XXX files. The program pdto3.f is also responsible for the creation of the sizeBP.dat file, which has an identical function to one used in the BP run. The input deck **dstgdto3** for this run is shown below:

```
&DTO3SYM INAST=28 /
```

After completing the pdto3.f stage, one then uses to pstg3r.f and pstgf.f to diagonalize the $(N+1)$-electron matrices and solve the asymptotic part of the problem, respectively. The input for these two parallel runs is identical to that of a BP run.

References

[1] Berrington K A, Eissner W B and Norrington P H 1995 *Comput. Phys. Commun.* **92** 290

[2] Burke V M, Burke P G and Scott N S 1992 *Comput. Phys. Commun.* **69** 76

[3] Norrington P H 2004 http://www.am.qub.ac.uk/DARC

[4] Chen G, Pradhan A K and Eissner W, 2003 *J. Phys. B* **36** 453

Appendix A : LS/Breit-Pauli Parallel R-Matrix Programs

- **pstg1r.f** - The bound-continuum and continuum-continuum integrals are distributed across the processors and each processor writes its own RKXXX.DAT file. Since the bound-continuum and continuum-continuum integrals dominate the execution time, the scaling is almost perfect; however, this type of parallelization requires the mapping of integrals within individual RKXXX.DAT files to their global position within the standard RK.DAT file – this is done in pstg2r.f.
- **pstg2r.f** - The calculations are distributed over the $SL\Pi$ partial waves by processor. Each processor writes a STG2HXXX.DAT file. There is no communication between processors. However, the efficiency of this code depends on the speed of communications between the mounted disk drive (where the RKXXX.DAT direct access files are stored) and the individual processors. On large supercomputers it is more efficient to run every $SL\Pi$ symmetry concurrently with one partial wave per processor; however, on smaller clusters with slower communications with the mounted disk drive, it is often more efficient to reduce the number of processors and run more partial waves on each processor.
- **pstg2.5.f** - This is a new code, which groups the $SL\Pi$ symmetries that contribute to a particular range of $J\Pi$ symmetries into a single binary STG2HJXXX file on a particular processor. Ideally, every $J\Pi$ symmetry should be calculated with one partial wave per processor. However, moderately large scale calculations can still be done on a local cluster, by grouping ranges of $J\Pi$ symmetries on a single processor.
- **precupd.f/pstgjk.f** - The transformations of the $(N + 1)$-electron matrix elements from pure LS coupling to intermediate coupling that are performed in this code are distributed by ranges of $J\Pi$ partial waves over the processors. Each processor writes a file RECUPHXXX file. In addition, the code determines the sizes of the Hamiltonian matrices for each symmetry and writes this information to a file **sizeBP.dat**; it is used to allocate the memory required for the construction and diagonalization of the $(N+1)$-electron Hamiltonian matrix within **pstg3r.f**.
- **pstg3r.f** - The parallel version of the matrix diagonalization code has been changed in several ways from its serial counterpart.
  - The parallel coding of **pstg3r.f** requires it to access matrix size information at the beginning of each run. Thus, for an LS run, in addition to multiple STG2HXXX.DAT files, the file sizeH.dat generated by **pstg2r.f** must be present. Similarly, for a BP or DARC run, in addition to multiple RECUPHXXX files, the file sizeBP.dat must be present.

  - All MPI broadcasting of the $(N+1)$-electron Hamiltonian matrix elements has been removed, as this was very slow for large cases. Each processor now reads these matrix elements directly from the STG2HXXX.DAT files in an *LS* case, and from the RECUPHXXX files for a BP case.
  - When the matrix size exceeds $\sqrt{2^{31}}$ ($\sim 46,340$), the indices which map a large Hamiltonian array into upper half of a symmetric matrix, exceed the integer*4 limit. Therefore the mapping subroutines **ijton, ntoij, locH** and **TAPERD** had to be upgraded to integer*8. The mapping functions have also been simplified.
  - A new method to form the surface amplitudes after the matrix diagonalization has been implemented. For large cases, the MPI command that was being used to send partial surface amplitudes to node 0 was completely overwhelming the communication system on even large massively parallel machines. Each processor now performs a partial sum of the surface amplitudes and sends that information to node zero, where the sum is completed.
  - The diagonalization routine PDSEYVX that was previously employed in some early versions of pstg3r.f (and also pstg3nx.f) has been phased out. This diagonalization routine should never be used since it only enforces orthogonality between the eigenvectors calculated on a particular processor. Therefore, with increased number of processors, the accuracy of the orthogonalization deteriorates.

Appendix B : Parallel Non-Exchange *R*-Matrix Programs

Work has been completed on the elimination of the major bottlenecks in the parallel non-exchange codes pstg2nx.f and pstg3nx.f.

- stg1nx.f - There is no parallelization required as the code executes in seconds.
- pstg2nx.f - The parallelization over the L and Π of the $(N+1)$-electron system and the spin S of the N-electron target. On massively parallel architectures, the scaling of pstg2nx.f is favorable due to faster communication between the processors and the mounted disk. However, due to the slower communication between the mounted disk and the processors on small clusters, peak performance is achieved using between 8 and 12 processors.
  - the array U(I,J) now has the correct index order required for the fastest calculation of the radial integrals in subroutine RK.
  - There are two options to divide the L values over the processors: (1) If MINLT and MAXLT variables *only* are present in the namelist, then the

range of L values is split as evenly as possible over the processors. If not divisible exactly, then the greatest range of L values is assigned to the partial waves with the highest L values. (2) With the second option an integer array namelist variable IWAVE is used to specifically control the distribution of the ranges of L values. This is useful because the $(N+1)$-electron continuum basis decreases as L increases; therefore it may be as fast to make calculations in the range $L = 45 - 50$ as for $L = 10$. This is explained in greater detail within the code – it is not the default option.

  – pstg2nx.f creates the file **sizeNX.dat** file listing the range of L values carried out on each processor. This file must be present when pstg3nx.f is run.

- pstg3nx - Only minor changes were required to read in the multiple radial integral files: RAD1.DATXXX, RAD2.DATXXX, RAD3.DATXXX. All MPI broadcasting of the $(N+1)$-electron matrix elements has been removed. Each processor now reads the angular coefficients and radial integrals; then using scratch files for temporary storage, the $(N+1)$-electron matrix elements are constructed. This code is slower than the exchange version (pstg3r.f) due to the increased computation needed to form the Hamiltonian matrix from the angular coefficients and radial integrals.

Appendix C : The Parallel DARC Suite of R-Matrix Programs

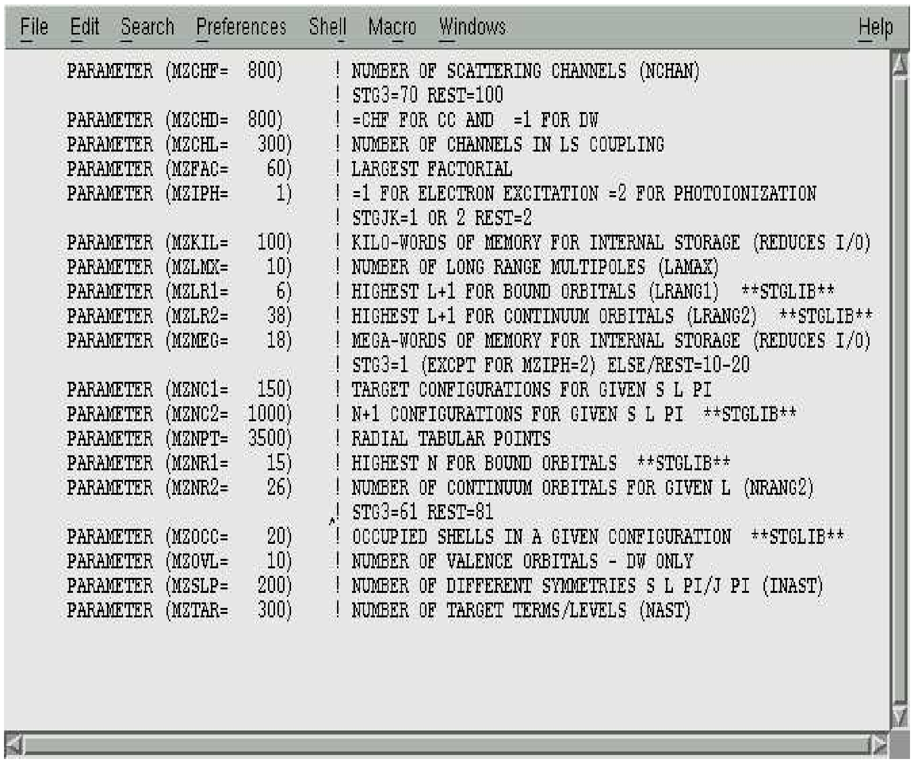
- stg1d_orbs.f - There is no parallelization required as the code executes in seconds.
- stg1d_ints.f - No parallelization was necessary; however, the integration subroutine INTRD2 has been made more efficient. The benefits are proportional to the size of NRANG2, and therefore, the number of times the above subroutine is called.
- pstg2d.f - This code was parallelized over the $J\Pi$ $(N+1)$-electron symmetries.
  - The order of the indexing was corrected in the formation of the Hamiltonian matrix in subroutines CMAT and CMATX
  - The efficiency of the extraction of electrostatic radial integrals from the array RKSTO in the subroutine FINCC has been improved. In addition, symmetry relationships have been employed to improve the speed of the matrix multiplies in this subroutine. These improvements had a significant effect since this subroutine is called many times.
- pdto3.f : This code reads the multiple passing files DSTG2.XXX, which contain the $(N+1)$-electron matrix elements and writes them to the RECUPHXXX files in a form that can be read by pstg3r.f.
- pstg3r.f - This program was modified to allow it to work with the DARC codes. It

distinguishes between a DARC calculation from a BP calculation by detecting a negative value of the variable LRANG2, which is communicated from the earlier codes using the passing files RECUPHXXX.

- pstgf.f - This program was also modified to allow it to work with a DARC calculation.

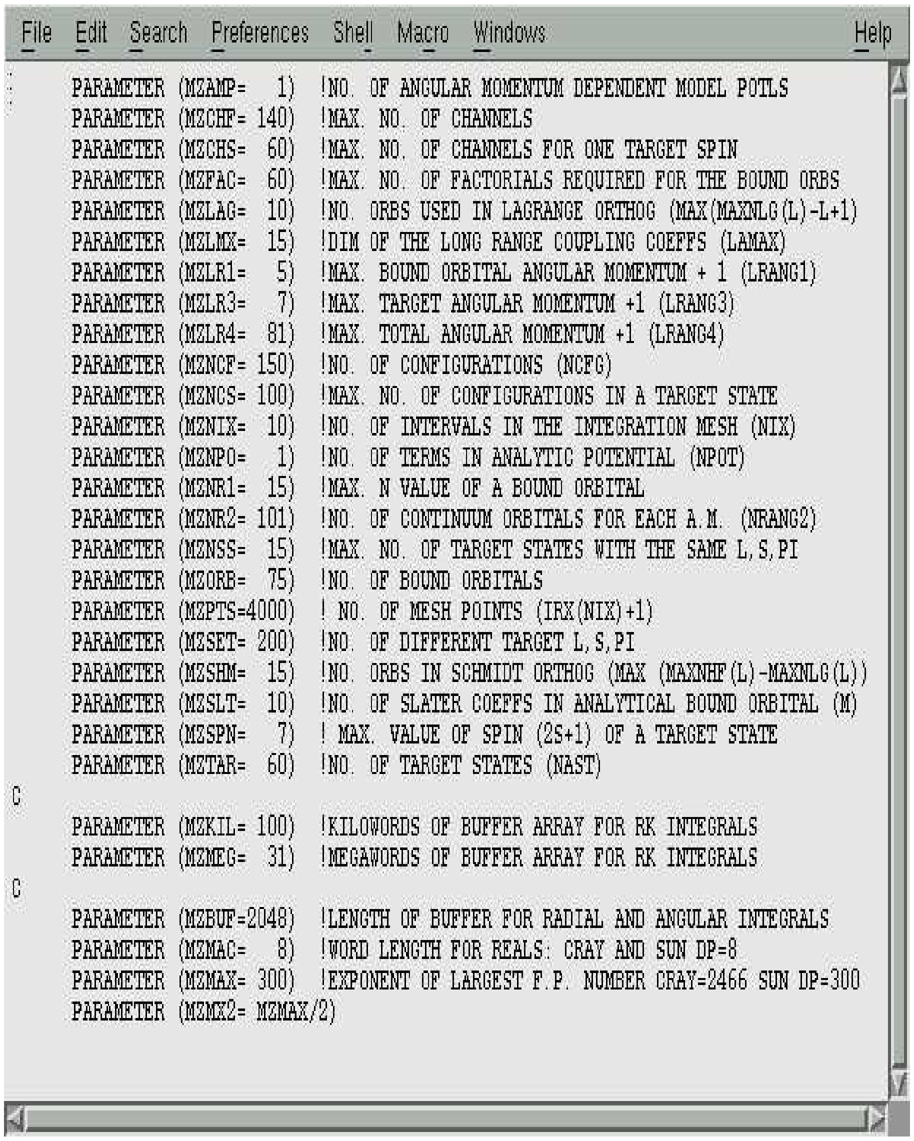
Appendix D : Sample PARAM Files

Shown below are sample PARAM files for the LS/BP inner-region codes, the non-exchange inner-region codes, the inner-region DARC codes, and the outer-region pstgf.f code.



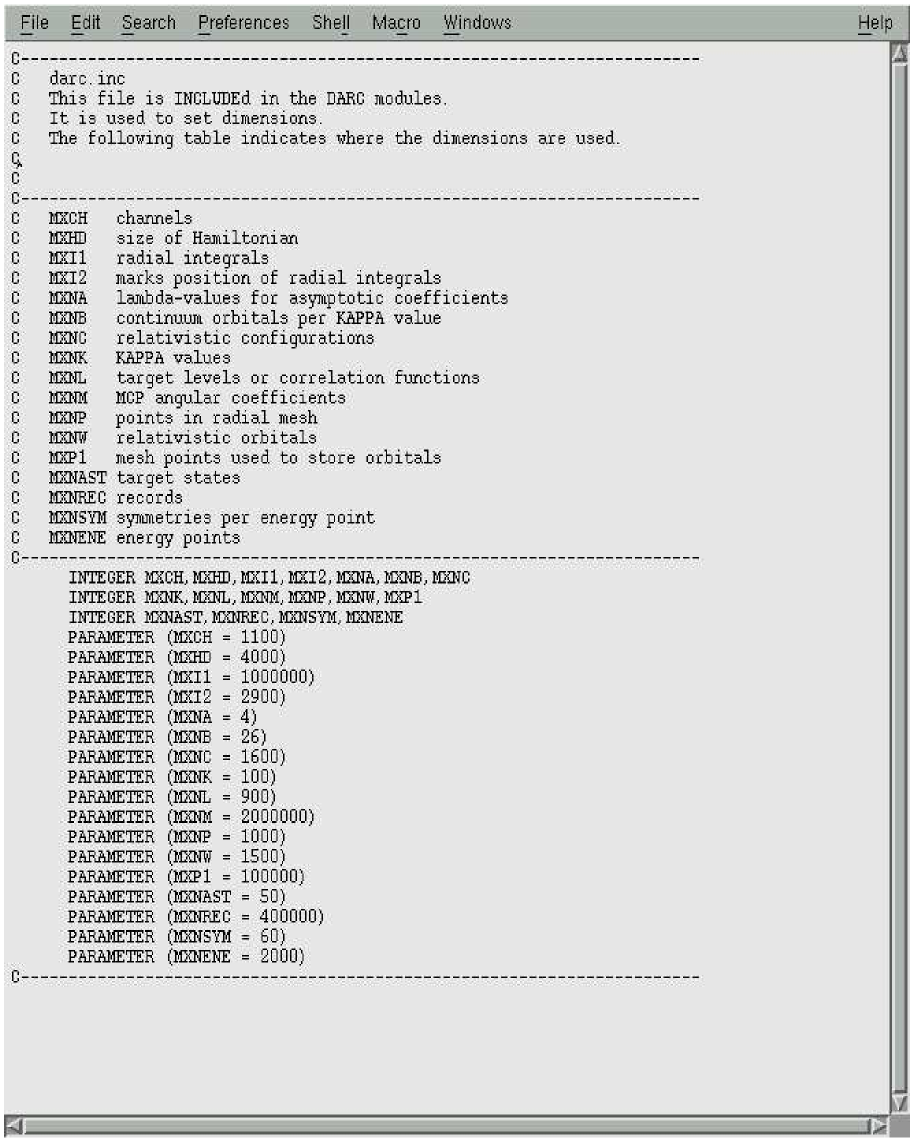
```
      PARAMETER (MZCHF=  800)     ! NUMBER OF SCATTERING CHANNELS (NCHAN)
                                  ! STG3=70 REST=100
      PARAMETER (MZCHD=  800)     ! =CHF FOR CC AND  =1 FOR DW
      PARAMETER (MZCHL=   300)    ! NUMBER OF CHANNELS IN LS COUPLING
      PARAMETER (MZFAC=    60)    ! LARGEST FACTORIAL
      PARAMETER (MZIPH=     1)    ! =1 FOR ELECTRON EXCITATION =2 FOR PHOTOIONIZATION
                                  ! STGJK=1 OR 2 REST=2
      PARAMETER (MZKIL=   100)    ! KILO-WORDS OF MEMORY FOR INTERNAL STORAGE (REDUCES I/O)
      PARAMETER (MZLMX=    10)    ! NUMBER OF LONG RANGE MULTIPOLES (LAMAX)
      PARAMETER (MZLR1=     6)    ! HIGHEST L+1 FOR BOUND ORBITALS (LRANG1)  **STGLIB**
      PARAMETER (MZLR2=    38)    ! HIGHEST L+1 FOR CONTINUUM ORBITALS (LRANG2)  **STGLIB**
      PARAMETER (MZMEG=    18)    ! MEGA-WORDS OF MEMORY FOR INTERNAL STORAGE (REDUCES I/O)
                                  ! STG3=1 (EXCPT FOR MZIPH=2) ELSE/REST=10-20
      PARAMETER (MZNC1=   150)    ! TARGET CONFIGURATIONS FOR GIVEN S L PI
      PARAMETER (MZNC2=  1000)    ! N+1 CONFIGURATIONS FOR GIVEN S L PI  **STGLIB**
      PARAMETER (MZNPT=  3500)    ! RADIAL TABULAR POINTS
      PARAMETER (MZNR1=    15)    ! HIGHEST N FOR BOUND ORBITALS  **STGLIB**
      PARAMETER (MZNR2=    26)    ! NUMBER OF CONTINUUM ORBITALS FOR GIVEN L (NRANG2)
                                  ! STG3=61 REST=81
      PARAMETER (MZOCC=    20)    ! OCCUPIED SHELLS IN A GIVEN CONFIGURATION  **STGLIB**
      PARAMETER (MZOVL=    10)    ! NUMBER OF VALENCE ORBITALS - DW ONLY
      PARAMETER (MZSLP=   200)    ! NUMBER OF DIFFERENT SYMMETRIES S L PI/J PI (INAST)
      PARAMETER (MZTAR=   300)    ! NUMBER OF TARGET TERMS/LEVELS (NAST)
```

Figure 6. An example of a PARAM file for the inner-region LS/BP codes



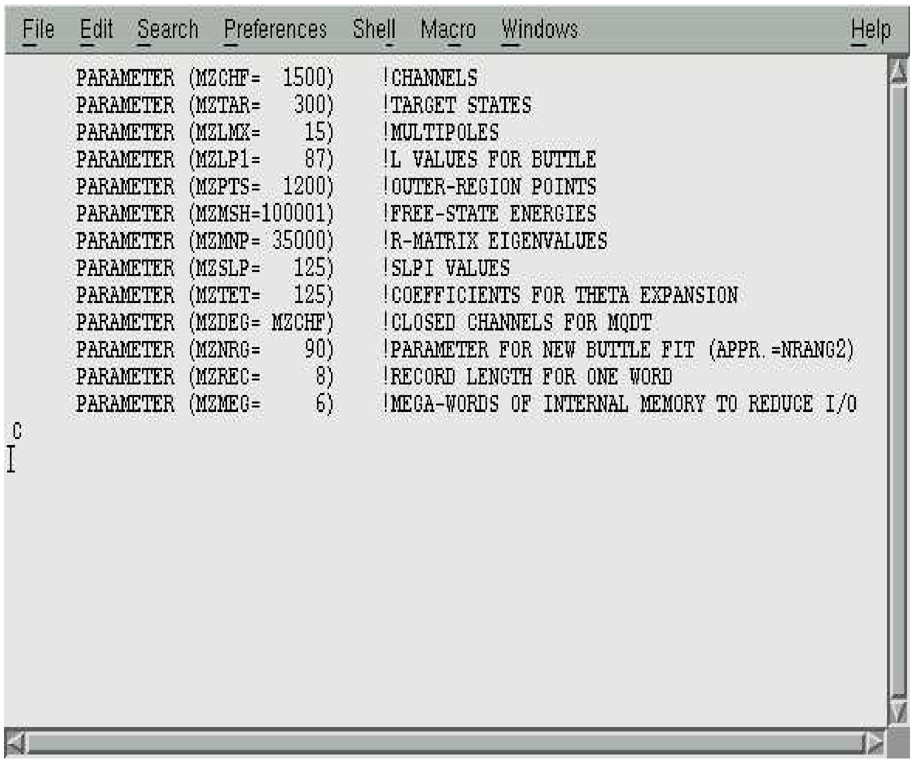
```
      PARAMETER (MZAMP=   1)  !NO. OF ANGULAR MOMENTUM DEPENDENT MODEL POTLS
      PARAMETER (MZCHF= 140)  !MAX. NO. OF CHANNELS
      PARAMETER (MZCHS=  60)  !MAX. NO. OF CHANNELS FOR ONE TARGET SPIN
      PARAMETER (MZFAC=  60)  !MAX. NO. OF FACTORIALS REQUIRED FOR THE BOUND ORBS
      PARAMETER (MZLAG=  10)  !NO. ORBS USED IN LAGRANGE ORTHOG (MAX(MAXNLG(L)-L+1)
      PARAMETER (MZLMX=  15)  !DIM OF THE LONG RANGE COUPLING COEFFS (LAMAX)
      PARAMETER (MZLR1=   5)  !MAX. BOUND ORBITAL ANGULAR MOMENTUM + 1 (LRANG1)
      PARAMETER (MZLR3=   7)  !MAX. TARGET ANGULAR MOMENTUM +1 (LRANG3)
      PARAMETER (MZLR4=  81)  !MAX. TOTAL ANGULAR MOMENTUM +1 (LRANG4)
      PARAMETER (MZNCF= 150)  !NO. OF CONFIGURATIONS (NCFG)
      PARAMETER (MZNCS= 100)  !MAX. NO. OF CONFIGURATIONS IN A TARGET STATE
      PARAMETER (MZNIX=  10)  !NO. OF INTERVALS IN THE INTEGRATION MESH (NIX)
      PARAMETER (MZNPO=   1)  !NO. OF TERMS IN ANALYTIC POTENTIAL (NPOT)
      PARAMETER (MZNR1=  15)  !MAX. N VALUE OF A BOUND ORBITAL
      PARAMETER (MZNR2= 101)  !NO. OF CONTINUUM ORBITALS FOR EACH A.M. (NRANG2)
      PARAMETER (MZNSS=  15)  !MAX. NO. OF TARGET STATES WITH THE SAME L,S,PI
      PARAMETER (MZORB=  75)  !NO. OF BOUND ORBITALS
      PARAMETER (MZPTS=4000)  ! NO. OF MESH POINTS (IRX(NIX)+1)
      PARAMETER (MZSET= 200)  !NO. OF DIFFERENT TARGET L,S,PI
      PARAMETER (MZSHM=  15)  !NO. ORBS IN SCHMIDT ORTHOG (MAX (MAXNHF(L)-MAXNLG(L))
      PARAMETER (MZSLT=  10)  !NO. OF SLATER COEFFS IN ANALYTICAL BOUND ORBITAL (M)
      PARAMETER (MZSPN=   7)  ! MAX. VALUE OF SPIN (2S+1) OF A TARGET STATE
      PARAMETER (MZTAR=  60)  !NO. OF TARGET STATES (NAST)
C
      PARAMETER (MZKIL= 100)  !KILOWORDS OF BUFFER ARRAY FOR RK INTEGRALS
      PARAMETER (MZMEG=  31)  !MEGAWORDS OF BUFFER ARRAY FOR RK INTEGRALS
C
      PARAMETER (MZBUF=2048)  !LENGTH OF BUFFER FOR RADIAL AND ANGULAR INTEGRALS
      PARAMETER (MZMAC=   8)  !WORD LENGTH FOR REALS: CRAY AND SUN DP=8
      PARAMETER (MZMAX= 300)  !EXPONENT OF LARGEST F.P. NUMBER CRAY=2466 SUN DP=300
      PARAMETER (MZMX2= MZMAX/2)
```

Figure 7. An example of a PARAM file for the inner-region non-exchange codes



```
C-----------------------------------------------------------------------
C   darc.inc
C   This file is INCLUDEd in the DARC modules.
C   It is used to set dimensions.
C   The following table indicates where the dimensions are used.
C
C
C-----------------------------------------------------------------------
C   MXCH   channels
C   MXHD   size of Hamiltonian
C   MXI1   radial integrals
C   MXI2   marks position of radial integrals
C   MXNA   lambda-values for asymptotic coefficients
C   MXNB   continuum orbitals per KAPPA value
C   MXNC   relativistic configurations
C   MXNK   KAPPA values
C   MXNL   target levels or correlation functions
C   MXNM   MCP angular coefficients
C   MXNP   points in radial mesh
C   MXNW   relativistic orbitals
C   MXP1   mesh points used to store orbitals
C   MXNAST target states
C   MXNREC records
C   MXNSYM symmetries per energy point
C   MXNENE energy points
C-----------------------------------------------------------------------
      INTEGER MXCH,MXHD,MXI1,MXI2,MXNA,MXNB,MXNC
      INTEGER MXNK,MXNL,MXNM,MXNP,MXNW,MXP1
      INTEGER MXNAST,MXNREC,MXNSYM,MXNENE
      PARAMETER (MXCH = 1100)
      PARAMETER (MXHD = 4000)
      PARAMETER (MXI1 = 1000000)
      PARAMETER (MXI2 = 2900)
      PARAMETER (MXNA = 4)
      PARAMETER (MXNB = 26)
      PARAMETER (MXNC = 1600)
      PARAMETER (MXNK = 100)
      PARAMETER (MXNL = 900)
      PARAMETER (MXNM = 2000000)
      PARAMETER (MXNP = 1000)
      PARAMETER (MXNW = 1500)
      PARAMETER (MXP1 = 100000)
      PARAMETER (MXNAST = 50)
      PARAMETER (MXNREC = 400000)
      PARAMETER (MXNSYM = 60)
      PARAMETER (MXNENE = 2000)
C-----------------------------------------------------------------------
```

Figure 8. An example of a PARAM file for the inner-region DARC codes



```
      PARAMETER (MZCHF=  1500)    !CHANNELS
      PARAMETER (MZTAR=   300)    !TARGET STATES
      PARAMETER (MZLMX=    15)    !MULTIPOLES
      PARAMETER (MZLP1=    87)    !L VALUES FOR BUTTLE
      PARAMETER (MZPTS=  1200)    !OUTER-REGION POINTS
      PARAMETER (MZMSH=100001)    !FREE-STATE ENERGIES
      PARAMETER (MZMNP= 35000)    !R-MATRIX EIGENVALUES
      PARAMETER (MZSLP=   125)    !SLPI VALUES
      PARAMETER (MZTET=   125)    !COEFFICIENTS FOR THETA EXPANSION
      PARAMETER (MZDEG= MZCHF)    !CLOSED CHANNELS FOR MQDT
      PARAMETER (MZNRG=    90)    !PARAMETER FOR NEW BUTTLE FIT (APPR.=NRANG2)
      PARAMETER (MZREC=     8)    !RECORD LENGTH FOR ONE WORD
      PARAMETER (MZMEG=     6)    !MEGA-WORDS OF INTERNAL MEMORY TO REDUCE I/O
C
```

Figure 9. An example ot a PARAM file for the outer region pstgf.f code