The Cowan code utilities: Description and Usage

Cormac McGuinness

Abstract

This document gives a brief description of the various Cowan code utilities that have been developed over the last two years. It describes their concept, method of operation, input and output. An attempt is made to describe how these utilites can be combined together with a good knowledge of shell programming to obtain almost automatic batch calculation of inner shell photoabsorption spectra.

Contents

1 The Cowan code

All of the utilites described in this document are only applicable when using the general atomic structure computer codes developed by Robert Cowan. It is assumed that the reader has some familiarity, not only with the use of the programs involved, but more specifically with the output generated by the last of the three principal codes, the RCG code. These utilities, developed in the UCD laboratory, are primarily concerned with electric dipole spectra from excitations of inner shell electrons. Almost all parameters of interest for the purposes of calculating inner shell photoabsorption spectra are contained within this sometimes large and complex output file. For further specific details about running

the code, the reader is referred to the Cowan code manuals. However, we do include a brief description of the principal components of the Cowan code[1]. These brief outlines also mention the standard filenames which the Cowan programs expect when executed.

1.1 The RCN code

The RCN code calculates one-electron radial wavefunctions (bound or free) for each of any number of specified electron configurations, using the Hartree-Fock or any of several more approximate methods. The principal output, for each configuration, consists of the center-of-gravity energy (E_{av}) of the configuration, and those radial Coulomb (F^k and G^k) and spin-orbit (ζ) integrals required to calculate the energy levels for that configuration.

The input file required by the code when executing, is an ascii text file named `in36` which contains descriptions of the electronic configurations for which the radial wavefunctions should be calculated. The results of these calculations are then stored for use by the RCN2 code in a binary file called `tape2n`. The RCN program also produces a text output file called `out36`.

1.2 The RCN2 code

The RCN2 program is an interface program that uses the output wave-functions from RCN (on a file called `tape2n`) to calculate the configuration-interaction Coulomb integrals (R^k) between each pair of interacting configurations, and the electric-dipole ($E^{(1)}$) and/or electric quadrupole ($E^{(2)}$) radial integrals between each appropriate pair of configurations.

There are two input files required for the execution of the RCN2 code. The first is an ascii text file named `in2` in which certain options can be set for the execution of the code. The most important of these is the ability to scale the radial integrals which are calculated. The second binary input file is called `tape2n` and contains the actual radial wavefunctions required to calculate the radial integrals. This file is produced by the RCN program. RCN2 prepares an output file called `out2ing` that (after being renamed `ing11`) serves as input to RCG. The RCN2 program also produces an ascii text output file called `out2`.

1.3 The RCG code

The third, RCG program, sets up energy matrices for each possible value of the total angular momentum **J**, diagonalizes each matrix to get eigenvalues (energy levels) and eigenvectors (multi-configuration, intermediate-coupling wavefunctions in various possible angular-momentum-coupling representations), and then computes $M^{(1)}$ (magnetic dipole), $E^{(2)}$, and/or $E^{(1)}$ radiation spectra, with wavelengths, oscillator strengths, radiative transition probabilities, and radiative lifetimes. Other options, when a continuum (free) electron is present, are photoionization cross-sections, autoionization transition probabilities, total lifetimes, branching ratios for autoionization, and plane-wave Born collision strengths.

[1]The following description comes from a `README` file which accompanied the source code for these programs. This gives a brief introduction to the codes and to their usage.

A number of files are associated with the RCG code and in normal usage we need only be concerned with two of these. The first is the required ascii text input file for the program which is called `ing11`. The contents of this file are data about the configurations involved in the calculation such as occupation numbers and configuration average energies, data about pairs of interacting configurations required for calculation of configuration-interaction (same parities) and for electromagnetic dipole transitions (different parities). Also included are options which allow the user to set the energy units used in the output and what coupling scheme the calculation should be performed in along with many other options.

The output file created by the RCG code is named `outg11`. This output file contains almost all relevant information about the calculation which has been performed. Two other output files are also produced if an option in the `ing11` file has indicated that the RCE program is to be used in conjunction with the RCG program. These are described in the next section.

1.4 The RCE code

When higher accuracy results are desired, the RCE program can be used to vary the various radial energy parameters E_{av}, F^k, G^k, ζ, and R^k to make a least-squares fit of experimental energy levels by an iterative procedure. The resulting least-squares-fit parameters can then be used to repeat the RCG calculation with the improved energy levels and (presumeably) wavefunctions. The input files required for RCE are produced when appropriate options are set in the input to the RCG program. These input files are called `tape2e` and `tape19`. A third input file is called `ine20`. The output produced by RCE is named `oute20`.

2 The Cowan shell script

Generally the first three Cowan programs (RCN/RCN2/RCG) are run in sequence with the output of each program forming an important part of the input to the next. To facilitate these codes being executed in an unbroken sequence a powerful shell script has been created. This Cowan shell script can be accessed by typing `cowan.sh` at the prompt on any of the Unix/Linux machines on which the code is installed[2]. Its usage is described here. The Cowan shell script also excutes the `shrink` code as the final part of the calculation sewuence. The `shrink` code is described in Section 3.

2.1 Filenames

Usage in UCD has resulted in the development of a certain nomenclature when choosing filenames both for input to and output from the Cowan code. The Cowan shell script developed here has followed this established usage when looking for and creating files during its execution.

The two vital files for any calculation are the input files for the RCN and the RCN2 code, called `in36` and `in2` respectively. By custom, when calculating the spectrum of, for instance, neutral strontium; then, the appropriate *in36* file

[2]As yet this shell script has not been optimised for use on Dos/Windows machines as it relies on the shell programming options available to users of the `bash` shell.

for that calculation would typically be renamed to *sr1*. The *in2* file required for the strontium calculation would then be renamed *sr1.in2*.

When the Cowan shell script is executed, by default it will prompt for the name of the input file for which the calculation is to be carried out. In the case above for a calculation of strontium neutral, then the name *sr1* would be supplied to the shell script. The shell script tests to see if that file exists, it will also test to see if the file *sr1.in2* exists. If either of these is missing then the script will exit with an error message.

What the shell script does is to copy the specified input file *sr1* to the file `in36` and also the input file *sr1.in2* to the file `in2`. These files being the desired inputs for the RCN and RCN2 programs respectively. As the calculation proceeds the various output files from the different programs are created and they are given new names; e.g. the text output file from the RCN program, `out36` will be renamed to *sr1.out36* after the calculation is over. Similarly, the text output from RCN2 will be renamed to *sr1.out2* while the output file which serves as the input to the RCG program will be adjusted and renamed *sr1.ing11*, ready for processing by the RCG program.

After the RCG program is finished, whose output file would be renamed *sr1.out*, the Shrink program will execute. This program produces many more files than those listed here, but all of these files will have a common root *sr1* but with different extensions to the filename. These files will be described in detail in Section 3.

2.2 Other tasks of the shell script

There are several other functions that it is useful for the shell script to perform automatically while executing the Cowan codes in their proper sequence. Largely these tasks are concerned with the housekeeping of all the files as described above. Other tasks however are to automatically prepare the input file for RCG, the `ing11` file so that all energies calculated in RCG are in terms of electron volts. Another task is to ensure that the RCG program has access to preformatted tapes with details of coefficients of fractional parentage on them. If these are not available, then the appropriate `cfp` decks are provided to create these on the fly. For further details please see the Cowan code manuals.

2.3 Usage of shell script

As mentioned above, when the shell script is executed, it prompts for the name of the input file for which the calculations are to be carried out. Alternatively, the shell script has also been programmed to accept a number of command line arguments, allowing the user to specify the input file for the calculation, whether to run only part of a calculation or to repeat a part where necessary. There are further options which allow different compiled versions of the code to be run which may be necessary for large or complex calculations.

The options available on the command line can be accessed by typing `cowan.sh -h`. This results in the following information being displayed on the screen:

```
cowan.sh [options] filename
        By default it runs Rcn/Rcn2/Rcg/Shrink in sequence for filename
        -h              Display usage of Cowan Code Shell script
        -m rcn          Runs Rcn/Rcn2 only
```

Filenames during execution of `cowan.sh`				
RCN				
sr1	→	`in36`		
		`out36`	→	*sr1.out36*
		`tape2n`	→	*sr1.tape2n*
RCN2				
sr1.in2	→	`in2`		
sr1.tape2n	→	`tape2n`		
		`out2`	→	*sr1.out2*
		`out2ing`	→	*sr1.ing11*
RCG				
sr1.ing11	→	`ing11`		
		`outg11`	→	*sr1.out*
Shrink				
sr1.out	→	`outg`		
		`outg.eav`	→	*sr1.eav*
		`outg.lab`	→	*sr1.lab*
		`outg.eig`	→	*sr1.eig*
		`outg.abseig`	→	*sr1.abseig*
		`outg.ls`	→	*sr1.ls*
		`outg.jj`	→	*sr1.jj*
		`outg.jjjk`	→	*sr1.jjjk*
		`outg.lslk`	→	*sr1.lslk*
		`outg.lsjk`	→	*sr1.lsjk*
		`outg.lsjlsj`	→	*sr1.lsjlsj*
		`outg.lsjlkj`	→	*sr1.lsjlkj*
		`outg.aa`	→	*sr1.aa*
		`outg.rad`	→	*sr1.rad*
		`outg.spec`	→	*sr1.spec*
		`outg.eo`	→	*sr1.eo*
		`outg.wo`	→	*sr1.wo*
		`outg.sorted`	→	*sr1.sorted*
		`outg.unsorted`	→	*sr1.unsorted*
		`outg.decays`	→	*sr1.decays*

Table 1: Table showing renaming of input and output files at each stage of a typical Cowan shell script calculation. Example shown: `cowan.sh` *sr1*.

```
-m rcg          Runs Rcg only
-m rce          Runs Rce only
-m shrink       Runs Shrink program only
-m clean        Removes temporary files
-d l            Runs Rcg using large dimensions
-d xl           Runs Rcg using extra large dimensions
-d xxl          Runs Rcg using extremely large dimensions
```

Therefore, to execute the RCN and RCN2 programs by themselves the following command would be typed in at the command line: `cowan.sh -m rcg` . The script will then prompt for the filename of the calculation. To run the large dimension version of the RCG code by itself the command to be used is: `cowan.sh -d l -m rcg` . Again, in this case, the script will prompt for the filename. The filename can also be explicitly specified on the command line as in the following case: `cowan.sh -d l -m rcg` *filename*.

As previously indicated, when the shell script starts running it checks to see if the specified files exist. Thus, when starting an RCG run it will check that there exists a *filename.ing11*, the appropriate input file for the RCG program. When starting to process the output of an RCG calculation through the Shrink program, it will first of all check for the existence of the RCG output *filename.out* before proceeding.

As indicated by the help information that can be displayed, the Cowan shell script can also clean the current directory of all the temporary files created during the running of the sequence of programs.

3 The Shrink program

The most useful program of all the Cowan utilities was the first to be written. It is known as the `shrink` program. It's concept is essentially to reduce the amount of information contained in the output of the Cowan code, specifically the output of the RCG program, into smaller more manageable chunks.

3.1 The RCG output file

The output of the RCG program can be briefly summarised into the following sequence of sections. They are listed in order of their approximate position within the ascii text output of the RCG program.

- Details of the configuration average energies and parities for each configuration included in the calculation. The configurations of each parity are assigned configuration numbers here.
- For each possible total angular momentum arising from the configurations, each state is labelled in terms of both the LS and jj angular-momentum-coupling schemes. The configuration numbers are augmented by serial numbers for each label within each group of possible total angular momenta.
- Various integrals are listed for each configuration and also for each pair of interacting configurations.

- The eigenvalues and the corresponding configuration serial numbers are listed for the lowest total angular momentum of the first parity.
- These are followed by the eigenvector coupling matrices, typically just those corresponding to the LS and jj coupling schemes.
- If continuum configurations have been entered into the calculation there may follow a matrix of A^a (or autoionisation) rates from the excited states through these decay channels.
- This sequence of energy eigenvalues, coupling matrices and rates is repeated for all total angular momenta of the first parity and then for all total angular momenta of the second parity.
- The radial overlap integrals for pairs of configurations are then listed.
- Lastly, the calculated spectrum with all the transition lines is listed.

The `shrink` program processes the RCG output file and as it encounters these features it write much of this data to new files. These files are (hopefully) easier to read/interpret than the complete file. Ideally, it will then be easier to access specific pieces of information from the calculation. In the following subsections we list all the various files that the `shrink` program produced and detail the information within each file. Where necessary the specific formatting of the file is discussed.

3.2 The files produced by the `shrink` program

When the `shrink` program is executed the RCG output file must have previously been renamed to `outg`. This file is then processed by the `shrink` program producing all of the sub-files listed below.

3.2.1 The *.eav* file

The *.eav* file is essentially the first part of the RCG output file listed above. It is formatted exactly as it is within the original RCG output. Each configuration is described via occupancy numbers, the number of electrons outside the core is listed, the parity is given (-1 for odd, 1 for even), the element label and the configuration label (as in the input file) are listed. Finally, the E_{av} in terms of the default energy unit is listed along with the kinetic energy of the continuum electron for continuum configurations.

3.2.2 The *.lab* file

The *.lab* file contains the angular-momentum coupling labels of all the states for all values of the total angular momenta. These labels are always given in LS and jj but also in other coupling schemes when desired. Their format vary with the number of open shells which are in the input configurations. Essentially the same formatting is used as in the RCG output file.

3.2.3 The *.eig* file

The *.eig* file is created when the `shrink` code encounters the eigenvalues which are listed for each angular momentum within the RCG output file. The format of the file created by the `shrink` program is of two columns of numbers, the eigenvalue and the configuration serial number multiplied by the parity. Each angular momentum grouping in the final *.eig* file is separated from the next by a label indicating the value of the total angular momentum for that group. All the eigenvalues are relative to the E_{av} of the first configuration specified in the input file.

3.2.4 The *.abseig* file

The *.abseig* file is very similar to the *.eig* file, except that the eigenvalue energies written to this file are no longer relative energies but absolute energies. In other words the relative energies, as before, but with the addition of the E_{av} of the first configuration in the input file. This can be useful when comparing energy level structure between different ions or even different elements.

3.2.5 The *.ls* and *.jj* files

The files *.ls* and *.jj* are the angular-momentum coupling matrices for the LS and jj coupling schemes as in the RCG output file. However, the storage requirements for these matrices, in terms of ascii text is much reduced and the presentation is clearer. This is done in two ways. Firstly, the coupling coefficients are all squared (though preserving their original sign) and expressed as percentages providing a clearer visual indication of configuration mixing. Secondly, as the coupling matrices are printed in blocks of 11 columns each, if any given sub-row of 11 columns does not contain a value above a threshold of 5% then that row is not copied to the new coupling matrix files reducing the storage required.

3.2.6 The *.jjjk, .lslk, .lsjk, .lsjlsj* and *.lsjlkj* files

These files are always created when the Shrink program runs but will have no information in them unless the RCG calculation was specifically enabled to print the coupling-matrices of these other angular-momentum coupling schemes or the entire calculation performed in one of these coupling schemes. Otherwise, the format of these files are exactly the same as the format of the *.ls* and *.jj* files.

3.2.7 The *.aa* file

The *.aa* file contains the autoionisation rate (A^a) matrix. For a given total angular momentum with n eigenvalues of which there are m discrete states the normal RCG output is a matrix of $n-m$ rows and n columns. All the rows correspond to decay channels but the columns correspond to all states, both discrete and continuum. In autoionisation calculations all the continuum-continuum matrix elements are zero, so these can be ignored and are not reproduced in the *.aa* file. In the *.aa* file the columns are labelled with the corresponding eigenvalue energy and the configuration serial number. The sum of the autoionisation rates

is displayed and converted from inverse seconds to milli-electron volts at the bottom of each column.

3.2.8 The *.rad* file

The *.rad* file contains the values of the radial overlap integrals between pairs of configurations of differing parities in the case of electric dipole transitions. The format is the same as that contained within the RCG output file.

3.2.9 The *.spec* file

The *.spec* file is the file created from the RCG output which lists all the electric dipole transitions which have been calculated. The format and content of each individual line is the same as in the RCG output but the header information is only included once, while all subsequent blank lines are avoided. Since this file is the most useful file produced by the `shrink` code, a brief description of the elements within the file is warranted.

The first part of the file is the ascii text header information which gives the configuration numbers and configuration labels of all the configurations used in the calculation grouped according to parity. Additionally there is a line of column headings describing the information recorded for each transition. This line should be retained if possible as several of the Cowan utilities use this line when reading in the information contained in this file.

The second part of the file is simply the list of all the transitions. For each transition the following information is recorded on each line: the number of the transition, the energy eigenvalue of the lower state, the configuration serial number of the lower state, the assigned LS label of the lower state, the energy eigenvalue of the upper state, the configuration serial number of the upper state, the assigned LS label of the upper state, the energy difference (or transition energy) between the states, the wavelength in angstroms, the value of `s/pmax**2`, the gf value, the logarithm to the base 10 of the gf value, the transition probability expressed in inverse seconds (the radiative lifetime), and a quantity described as the cancellation factor.

3.2.10 The *.eo* and *.wo* files

Both the *.eo* and the *.wo* files are simple listings of energy against oscillator strength or wavelength against oscillator strength respectively. The only text contained in the file is in the first line which serves to identify the columns.

3.2.11 The *.sorted* and *.unsorted* files

As the `shrink` program processes the RCG output file, it retains information in memory regarding all of the discrete states that it encounters within the file. Upon reaching the end of the RCG output file it writes all of this information to two files. These files are the *.unsorted* and *.sorted* files. The *.unsorted* file is a listing of all the states in the order in which they were encountered while the *.sorted* file sorts all of these states in order of increasing energy eigenvalues.

The contents of these files are simply listings of all the states giving their relative eigenvalues, their configuration serial numbers multiplied by their parity, their total angular momentum, and then a brief summary of their eigenvector

composition. The four leading components of their eigenvector composition are included giving the percentage composition followed by the state serial number, the state coupling label (in the coupling in which the calculation is performed) and the configuration label of that component.

3.2.12 The *.decays* file

Similar to the *.sorted* and the *.unsorted* files, the `shrink` program also retains information about all the calculated autoionisation widths contained within the RCG output file. This information is then written to the *.decays* file. The format of this file is that for each group of total angular momenta the energy eigenvalues of the decaying states, their configuration serial numbers multiplied by their parity, their total angular momenta, their configuration label, their LS coupling label, their state serial number are all printed on each row. Furthermore, the configuration label, the angular-momentum-coupling label in which the calculation was performed as well as the state serial number are written, with the last item being the decay width (or lifetime) of the state in milli-electron volts. The majority of this information is used by one of the Cowan utilities which will be described later.

3.3 Executing the `shrink` program

The execution of the `shrink` program is normally undertaken immediately after the sequence of Cowan programs (RCN/RCN2/RCG). This is automatically catered for in the Cowan shell script. Should it be necessary to run the `shrink` program on it's own at a later date, the Cowan shell script may be used to do this. This can be achieved by typing the command `cowan.sh -m shrink` *filename*. The usual file renaming occurs before and after the `shrink` program executes.

4 The Cowan utilities

The majority of the remainder of the Cowan utilities are much smaller and less complex programs than the `shrink` utility. In general they only operate on the output of the `shrink` utility and not on the output of Cowans RCG progam directly. The operation of the individual utilities is described below with some notes on the coding and usage of each of them. The majority of these programs have specific filenames hard-coded within them and so shell scripts have been developed to make their operation more useful. If the user wishes to use these programs without the shell scripts a list of the required input files and the subsequent output files produced is included in Table 2.

4.1 The `sortbyj` utility

The `sortbyj` utility acts on the *.spec* output of the `shrink` program. The *.spec* output is essentially the list of electromagnetic transitions calculated for the input configurations. Typically these are ordered in terms of increasing photon energy, however other options to the RCG code may result in them being ordered differently. For the program to execute the desired *.spec* file should be given the name *sortj.spec*.

Utility	Required Input Files	Output Files
sortbyj	sortj.spec	sortj.sorted.spec
sortbylower	sortlower.spec	sortlower.sorted.spec
splitspec	outg.spec	outg.*energy*.spec
spectemp	outg.spec	outg.state outg.dist outg.temp.spec
subgamma	subg.spec subg.decays	subg.sub.spec subg.not.spec
ewop	outg.spec	outg.ewop outg.wo outg.eo

Table 2: Cowan utilities that require particular input files

Regardless of the order in which the transitions are listed in the *.spec* file the `sortbyj` utility orders these transitions so that all transitions from the ground state with the lowest total angular momentum are grouped together at the beginning of the new sorted *.spec* file. The next grouping consists of all the transitions from the ground state with the next lowest total angular momentum. The transitions are further sorted within each of these groupings according to the total angular momentum of the upper states and again according to the photon energy of the transition. The new sorted *.spec* file is then written to a new file with the name *sortj.sorted.spec*.

For ease of use, there is a simple shell script which can execute the `sortbyj` program without the need for the user to rename files. This shell script is called `sortbyj.sh`. Help on how to use it can be accessed by typing `sortbyj.sh -h` which results in the following being displayed:

```
Usage: sortbyj.sh [options] sortedfile
        -h      Help screen
        -s file Specify a particular spectrum file to sort
        Extensions of '.spec' are assumed
```

In general the script is executed by typing: `sortbyj.sh -s` *filename newfilename.spec* .

4.2 The `sortbylower` utility

The `sortbylower` program is very similar to the `sortbyj` program. In this case the transitions in the *.spec* file are sorted according to increasing energy of the lower states of all the transitions. Thus a transition originating from a lower state with a particular energy is grouped together with similar transitions and sorted within that group according to photon energy.

The name of the input *.spec* file required by the `sortbylower` program is *sortlower.spec*. After processing the new sorted file is written to a file named *sortlower.sorted.spec*. For ease of use there is a shell script called `sortbylower.sh` which can be executed which handles the renaming of the files. The help screen for the shell script can be called up by typing `sortbylower.sh -h` which prints the following:

```
Usage: sortbylower.sh [options] sortedfile
        -h      Help screen
        -s file Specify a particular spectrum file to sort
        Extensions of '.spec' are assumed
```

In general the script is executed by typing: `sortbylower.sh -s` *filename newfilename.spec* .

4.3 The `splitspec` utility

The `splitspec` utility operates by reading in a *.spec* file and splitting it into many other *.spec* files. Each of these new *.spec* files corresponds to transitions which originate from a unique lower state. Thus, if there are n unique lower states in the original *.spec* file then n new *.spec* files will be created. Each of these new *.spec* files will retain the text header information of the original file. This header information is important as it is used in a number of the other Cowan utilities which process the *.spec* files.

A shell script called `splitspec.sh` has been written to take advantage of this utility. Information on the use of this shell script is obtained by typing `splitspec.sh -h` . This results in the following information being displayed:

```
Usage: splitspec.sh [options] inputfile
        -h      Help screen
        -s      Specify that a stick plot is to be calculated
        -c      Specify a cross-section plot to be calculated
        Extensions of '.spec' are assumed
```

Optionally the script can automatically call the `spec2plt` utility and produce a plot file from each new individual *.spec* file. These automatically generated plots are either stick plots of oscillator stengths or plots of the photoabsorption cross section arsing from each lower state.

4.4 The `spec2plt` utility

The `spec2plt` utility is almost the only Cowan utility which requires user input while it is running. This utility enables plots to be created from one or many *.spec* files. As mentioned previously the *.spec* files to be used by the `spec2plt` program must retain the column header information usually found in these files. This enables the `spec2plt` utility to properly interpret the dat within the file. No shell script is necessary to run this program as all the filenames are requested by the program. When the program is executed the following options appear:

```
 Spectrum plotting routine based on Cowan output
 Please choose plot type:
        0 = Stick plot of oscillator strengths  =.stk
        1 = Constant width Gaussian profiles    =.prn
        2 = Constant width Lorentzian profiles  =.prn
        3 = Cross section (widths needed)       =.prn
```

Selection of each option is by typing in the appropriate number. A brief description of the methods used to create each of these plots is given here.

4.4.1 Stick plots of oscillator strengths

A stick plot of oscillator strengths can be produced very easily. The options immediately available are whether to use the gf values, f_{abs} values or the f_{emis} values. The program prompts you for this selection thus:

```
You have chosen plot type:  0
Please choose oscillator strengths to use;
       0 = gfs
       1 = f_absorption
       2 = f_emission
```

The absorption oscillator strength f_{abs} values are equal to $gf/g_{lower} = gf/(2j_{lower}+1)$ while the emission oscillator strength f_{emis} values equal $gf/g_{upper} = gf/(2j_{upper}+1)$. All the oscillator strengths in given wavelength intervals (separated by 0.005Å) are summed together and the array is written to a file. By default the file is written in the form of energy versus summed intensity. To conserve storage space, as the file is written as ascii text, only those points which have summed intensities above the 10^{-8} threshold are written out to the file.

More than one *.spec* file can be accumulated in the resultant graph, and the program prompts for the number of files to read in. Following this the user is prompted as to whether they want to apply an energy shift to any of the spectra. If the answer is `y` then the user is prompted for a shift for each file as it is read in. This energy shift should be in electron volts. Furthermore, a weighting coefficient can be applied to the oscillator strengths of each *.spec* file as it is read in to the program. The program asks whether weighting coefficients are required. These additions to the program serve to make it as flexible as possible for producing graphs of spectra from, for instance, different ion stages.

When the `spec2plt` program prompts for the name of the input *.spec* file, it is assumed that there is an additional *.spec* extension at the end of the filename which you input to the program. Finally, when the plot is complete, the output plot file must be given a name. An extension of *.stk* or *.prn* is appended to the name you supply depending upon the option chosen at the beginning of the program. When producing stick plots of oscillator strengths it always appends the *.stk* extension.

4.4.2 Constant width Gaussian profiles

When this option is selected each spectral transition can be assigned the same spectral width and given a Gaussian line profile of that width. Again there is the choice as to which oscillator strength is desired for use in the plot. The program prompts for a width in electron volts which is then given the line profiles of all the transitions. The formula used for the Gaussian profile is:

$$\sigma(\omega) = exp\left(- \frac{(\omega - \omega_0)^2}{\frac{\Gamma_{const}}{2}^2} \right)$$

where ω_0 is the center frequency (energy) of the transition and Γ_{const} is the assigned width of all of the transitions. For each transition, this line profile is multiplied by the chosen oscillator strength, whether gf, f_{abs} or f_{emis} and the resultant intensity at any wavelength is the sum of all contributions at that wavelength. To reduce processing time, all contributions lower than 10^{-4} of the

height of each line profile are ignored. When using more than one *.spec* file as input both shifting and weighting coefficients may be specified.

4.4.3 Constant width Lorentzian profiles

This option is similar to the last with the exception that a Lorentzian line profile is used with the constant width specified. The form of oscillator strength to be used is chosen as is the width selected for all of the transitions. The formula used for the Lorentzian line profile is as follows:

$$\sigma(\omega) = \frac{\frac{\Gamma_{const}}{2}^2}{(\omega - \omega_0)^2 + \frac{\Gamma_{const}}{2}^2}$$

where ω_0 is the frequency of the transition and Γ_{const} the selected width of all the transitions. As above, the line profiles are multiplied by the form of oscillator strength chosen and shifting and weighting coefficients can be applied when more than one *.spec* is used.

4.4.4 Cross section

If the user chooses to create a cross section plot then it is assumed that the widths of all the spectral lines (or equivalently the lifetimes of the upper states) are contained within the *.spec* file. These are typically contained in the column marked by `gA(sec-1)` in the *.spec* file. From the normal RCG output these are simply the calculated radiative lifetimes of each state. When these widths are read in they are converted automatically into widths in units of electron volts within the program.

Where autoionisation calculations have been performed for the excited configurations then by using the `subgamma` utility these values may be replaced by the calculated decay widths expressed in milli-electron volts. When such a substitution has taken place then the header of the *.spec* file will contain `gA( meV )` instead of the label as in the previous instance. The `spec2plt` program will understand this rewritten header and read in the lifetimes in the appropriate units.

The formula used in conjunction with the calculated decay information assumes the line to be an isolated transition resulting in a Lorentzian shape of the photoabsorption cross section. This formula can be expressed by:

$$\sigma_{abs}(\omega) = 109.7617 * (f_{abs})_0 \frac{\frac{\Gamma_0}{2\pi}}{(\omega - \omega_0)^2 + \frac{\Gamma_0}{2}^2}$$

where $(f_{abs})_0$ is the absorption oscillator strength of the transition, ω_0 is the energy of the transition and Γ_0 is the width of the transition (or lifetime of the excited state). The numerical factor results in the cross section of the line being in units of megabarns when all the energies and decay widths are specified in electron volts.

The summed contributions from all transitions are combined and written to the file specified. As with the other methods, shifting and weighting coefficients can be applied to combine separate *.spec* files into one plot. The data in the file should then be two columns of photon energy and summed cross section (in megabarns) respectively.

4.5 The `spectemp` utility

The `spectemp` utility is designed to alter the *.spec* file to mimic the effect of temperature distributions among the lower states of a photoabsorption spectrum. The gf-values of all the transitions are altered to reflect the weighting in the population distribution that is associated with the lower state of each transition. Ratios of the upper and lower state populations are calculated between the ground state (the lowest of the lower states) and all other lower states for electric dipole transitions according to the Boltzmann population distribution equation:

$$\frac{N_i}{N_j} = \frac{g_i}{g_j} exp(-\frac{E_i - E_j}{kT})$$

where g_i and g_j are the degeneracy of the states, $2j_i+1$ and $2j_j+1$ respectively, E_i and E_j are the energies of states i and j respectively and kT is the Boltzmann temperature of the population distribution. This equation gives the ratios of the populations of the lower states to the population of the ground state. The weights are then determined by requiring that the sum of the weights of all the lower states is unity while still keeping the ratios between the lower states according to the equation above.

This information on the population distribution is recorded in two separate files. The first file `outg.dist` records the weighting distribution for the given Boltzmann temperature among all the lower states according to the unique energies of each state. The second file `outg.state` displays and sums all the contributions to the total weighting for a particular LS term. Within each LS term the ratios between the levels are of course determined by the Boltzmann distribution.

When the `spectemp` program is executed the user is prompted for the Boltzmann temperature to be used for the population distribution. This temperature is then used to produce the `outg.dist` and the `outg.state` files described above. The weightings for each set of transitions originating from a particular lower state (as given within the `outg.dist` file) are then multiplied by the gf values for the transitions originating from that state. This is repeated for all the lower states until it has been applied to all the transitions in the file. A new *.spec* file is then written with the name `outg.temp.spec` with the altered gf values. This new *.spec* file can then be used for plotting the temperature affected spectrum.

At present, no attempt is made to allow for calculating any temperature distribution between different ionisation stages, as may be observed simultaneously within a laser produced plasma spectrum. However, a temperature affected spectrum could be calculated separately for each ionisation stage assuming the same Boltzmann temperature and then from knowledge of plasma dynamics or estimates of the plasma electron temperature a guess for the population ratio of the different ion stages can be made. Using weighting coefficients from such an etimate the *.spec* files for the differing ionisation stages may be combined together using the `spec2plt` utility.

A shell script has also been devised for the use of this utility. The name of the shell script is `spectemps.sh`. This shell script enables the user to specify very simply the temperature for the spectral distribution. It's help information can be accessed by typing `spectemps.sh -h` which results in the following display:

```
Usage: spectemps.sh [options] inputfile
```

```
-h      Help screen
-t real Specify temperature(s) to calculate a spectrum for
-s      Specify that a stick plot is to be calculated
-c      Specify a cross-section plot to be calculated
Extensions of '.spec' are assumed
```

The new *.spec* file can easily be passed to the `spec2plt` program as indicated by the available options to either give a stick plot spectrum or a summed cross section spectrum valid for the given Boltzmann temperature. Note that a range of different temperatures can be calculated. These will be prompted for by the shell script which must be supplied with real numbers separated by spaces. To specify several temperatures on the command line, the temperatures must be enclosed in double quotes (‘‘) following the `-t` option. The temperature supplied must be in the same energy units as the levels calculated and listed in the *.spec* file. This is usually electron volts.

4.6 The `subgamma` utility

The `subgamma` program is designed to facilitate the easy creation of cross section plots by allowing similar calculations for the autionisation decay widths and for the electric dipole spectrum to be combined together. Essentially the output of the `subgamma` program is a new *.spec* which differs from the input *.spec* file in that the data given in the columns marked `gA(sec-1)` is no longer simply the radiative lifetime but the sum of the radiative lifetime and the calculated auoionisation decay rate for that upper state.

The `subgamma` program requires that information be available on the lifetime of each upper state in the *.spec* file to be processed. This information on the lifetimes of the upper states is usually obtained from one or more *.decays* files. Contained within each decay file is the necessary information for the identification of any given upper state. These are the relative energy eigenvalue of the upper state, the total angular momentum of that state and the LS identification of that state.

All the information from the *.decays* file is read in and stored. As the *.spec* file is processed matches are sought for the upper state of each transition with the states which were read in from the *.decays* file. For an exact match to be found the LS label must be the same, the energy eigenvalue must be the same and the total angular momentum of the states must be the same. When a match occurs then the information on the autoionisation decay rate of that state is added to the radiative lifetime information of that state and is written to a new *.spec* file. the name of this new file is `subg.sub.spec`.

If an **exact** match is not found, then the criteria for a match are gradually relaxed. The LS label no longer needs to be an exact match if a decay state with the same energy eigenvalue exists. If there is no exact match in energy, then the decay state which is nearest in energy is used instead, provided the energy difference between the upper state and the decay state is not larger than one electron volt. At all times the total angular momenta of both the decay state and the upper state must be the same. If there are no decay states found for a particular upper state then that transition is written to a different file to those for which matches were found. the name of the file containing the transitions to the unmatched upper states is `subg.not.spec`.

If possible the calculation which produces the *.decays* file should have the exact same configuration interaction in the upper state as there is in the calculation which produced the *.spec* file. Furthermore, to have the relative energy eigenvalues for each state to be the same in both the autoionisation calculation (*.decays* file) and the the calculation of the spectrum (*.spec* file), then the first lower state configuration in each calculation must be the same. This is because all energy eigenvalues (including those listed in the *.spec file*) are measured relative to the configuration average energy (E_{av}) of the first configuration in the input file.

The `subgamma` program can be called in a more user-friendly manner by executing the `subspec.sh` shell script. Help on this script is available by typing `subspec.sh -h` which results in the following display:

```
Usage: subspec.sh [options] outputfile
        -h      Help screen
        -f file Specify similar spectrum and decay files
        -s file Specify a particular spectrum file
        -d file Specify a particular decay file
        -n #    Specify number of decay files (Prompted for names)
        Extensions of '.spec' and '.decays' are assumed
```

This shell script allows the use of *.spec* and *.decays* files which have been calculated simultaneously or an amalgam of differing *.decays* files calculated for the individual upper configurations contained within the *.spec* file either combined within one file or in a number of files.

4.7 The `ewop` utility

The `ewop` program is designed to facilitate an earlier MS-Windows version of the `spec2plt` program that was known as the `Cowan Plotter`. The input to the `Cowan Plotter` program required that the *.spec* file be reduced into a file of four columns, the energy, the wavelength, the oscillator strength and the transition probability (hence *.ewop*). This `ewop` program takes in a *.spec* file and produces from it an *.ewop* file and also an *.eo* and a *.wo* file. The last two are useful if you the *.spec* file produced has been passed through the `spectemp` program for instance which alters the oscillator strengths in the program.

A small shell script called `ewop.sh` has been written to facilitate the use of the `ewop` program. The name of the *.spec* file is supplied to the `ewop.sh` script on the command line and the corresponding *.ewop*, *.eo* and *.wo* files are produced having the same filename, but of course with differing file extensions.

4.8 The `convolve` utility

The `convolve` program has been designed to be a useful way of finalising synthetic spectra created by calculations employing the Cowan code and the various aforementioned utilities. Since most experimental observations are made with spectroscopic instruments which do not have infinite resolving powers it is frequently desirable to either deconvolve the observed spectra or to convolve the calculated spectra. In practice, it is easier to convolve the calculated spectra to generate synthetic spectra for comparison as very narrow features in comparison to the instrumental function cannot be deconvolved successfully.

The `convolve` program allows the user to take a plot file (usually generated by `spec2plt` and with a *.prn* file extension) and convolve the spectrum within the file with a given instrumental function. Several different instrumental functions can be specified. These are

- Instrumental function of constant width (Gaussian lineshape)
- Instrumental function of constant width (Lorentzian lineshape)
- Instrumental function of varying width (Gaussian)
- Instrumental function of varying width (Lorentzian lineshape)

where the widths of the varying instrumental functions are simply determined as a factor of the photon energy (or wavelength) e.g. $\Delta E/E = 1000$ ($\Delta\lambda/\lambda) = 1000$). This would be more applicable to most spectroscopic instruments when considering very wide photon energy ranges than an instrumental function of constant width.

Unfortunately, at the time of writing this document, the `convolve` program is not available for general usage. In other words it hasn't been satisfactorilty completed yet.

4.9 The `addplots` utility

The `addplots` utility is the last and, so far, least used of the Cowan utilities described here. It is included to complete the desired range of abilites of the set of Cowan utilities. In part, its original specification has been superseded by the `spec2plt` program. It was originally envisaged as a utility to add separately calculated plots together with shifts and weights. However, the `spec2plt` program can achieve this by itself.

The input to the `addplots` utility is driven directly from the keyboard as is the case of the `spec2plt` program. The user is prompted for the number of plots to be combined, for shifting and weighting coefficients as necessary and for the complete filename of each plot, as well as the complete filename of the desired output file.

5 Combining utilites in a batch job

The utilities described above can, in general, through use of the shell scripts provided or through use of the properties of the unix shell environment can easily be run as single command lines.

5.1 Producing a plot file

The most useful result of a batch job is a tangible plot that can be compared visually with either other calculations or perhaps experimental results. The utilities described in the previous section are not graphical programs as it would be a case of "reinventing the wheel" to produce a good graphics package for displaying the results of calculations. Instead, the utilities `spec2plt`, the `convolve` and `addplots` produce output files which should be suitable for almost any good plotting package.

These plot files are normally produced from processing *.spec* files using the interactive `spec2plt` program. This has a simple interactive interface which prompts the user for options and filenames. This interface is surprisingly easy to interface into a batch script. The only thing necessary to drive the program is a simple way of interacting with these choices. In unix, this can be easily done, especially when using the `bash` command shell.

When using the `bash` shell, typing the command: `echo "`*text*`"` results in

text

appearing on the console. The `echo` command also understands C-style escape codes when given the `-e` option and so typing `echo -e "`*text*`\n`*more text*`"` results in:

text
more text

appearing on the console. In this case the `\n` has resulted in a newline being inserted between the two pieces of text. The output of the `echo` command may be piped through the `spec2plt` program with a line produced for each question that the `spec2plt` program requires an input for. This is done by using the unix pipe command `|`.

A simple example is where a stick plot of oscillator strengths is required from the *.spec* file *sr1.spec*. This is how it can be done:

`bash$ echo -e "0\n0\n1\n`*sr1*`\n`*sr1*`\n" | spec2plt`

This example tells the `spec2plt` program (in response to its normal queries) to produce a stick plot (option `0`) using gf values (option `0`) and one *.spec* file (number of files `1`) which has the name *sr1.spec* and to save the plot file in the file *sr1.prn*. This can be equally applied to calculating a plot of cross-section against photon energy for the same file by typing the command:

`bash$ echo -e "3\n1\n`*sr1*`\n`*sr1*`\n" | spec2plt`

Note that in this case one less input is required as there is no prompt for the type of oscillator strength to be used when producing a cross section plot.

5.2 Producing and printing a graph

The packages that could be used for plotting results of calculations range from `MathCad`, and `matlab` to `gnuplot`, but this author's favourite is the `xmgr` plotting package. This software package is available from the internet address http://plasma-gate.weizmann.ac.il/Xmgr/ and is currently only available on unix systems. The overwhelming advantage of `xmgr` is that it can be called from batch files and still produce high quality postscript plots using relatively simple single line commands.

The full point and click interface for `xmgr` can be accessed only through X-windows. To view a plot file using the program it can be simply started by typing `xmgr` *sr1.prn* which loads up the plot file *sr1.prn*. When viewing stick plots it is necessary to change the symbol type shown in the plot to the `Impulse at X` type and the line type to `None`. This can be done through the user interface or on the command line through typing the command: `xmgr` *sr1.stk* `-pexec "s0 symbol 12" -pexec "s0 linestyle 0"` . When executing `xmgr` in a batch mode the command used in batch shell files is `grbatch`. Exchanging `grbatch` for `xmgr` in the command above results in the graph being produced on the default printer attached to the computer. To produce a postscript file in landscape mode the command to use in the batch file would be: `grbatch`

sr1.prn `-landscape -printfile` *sr1.ps*. Many other command line options for `xmgr/grbatch` are available which can be used to set other plot parameters. In fact a parameter file, specifying labels/titles/axes/etc can be saved from an existing plot and reused from the command line quite easily. Full online help is available through the user interface or alternatively on the internet.

5.3 Using the spectrum sorting and splitting utilities

To sort the transitions contained within the *.spec* files in a batch command is quite straightforward. A command of the type: `sortbylower.sh -s` *sr1 sr1.sorted* sorts the transitions listed in the file *sr1.spec* according to the energy of the lower state and writes it to the new file *sr1.sorted.spec*. The same style of command can be used with the `sortbyj.sh` shell script.

To use the spectrum splitting utility shell script `splitspec.sh` on the command line is just as simple. The command: `splitspec.sh -s -c` *sr1* splits the file *sr1.spec* into as many different *.spec* files as there are distinct lower states in the original file. Stick plots and cross sections are then produced for each of these files.

5.4 Using the width substitution utility

For simple automatic autoionisation scripts, where all transitions under consideration are to one excited state which then decays, the use of the `subspec.sh` utility is very simple. If the calculation is named *sr1*, then the files *sr1.decays* and *sr1.spec* have been calculated. To get the `subspec.sh` to substitute the calculated decay widths of the upper states into the list of transitions and create a new list of transitions the command would be: `subspec.sh -s` *sr1* `-d` *sr1 sr1.sub* The new file called *sr1.sub.spec* can then be used to create plots of photoabsorption cross section. Alternatively, the same effect can be achieved by typing: `subspec.sh -f` *sr1* which assumes that both the *.decays* and the *.spec* files have the same root name of *sr1*, and automatically generates the new file *sr1.sub.spec*.

5.5 Putting it all together

To put a batch calculation together in a shell script in unix, just create a file and put the commands in line by line. To make the file executable the properties on it have to be modified using: `chmod +x` *filename* and, further, to have the contents of the file executed by the `bash` shell the first line of the script must be `#!`*/bin/bash*.

6 Limitations of the Cowan utilities

The Cowan utilities and the Cowan code all have limitations as to the largest calculation that can be achieved using them. Largely, these limitations are imposed due to memory constraints on the machine which is running the program and are thus determined at compilation time by use of dimensional parameters within the Fortran source code. This problem is largely due to the difficulty in dynamically sizing arrays within the Fortran language. The various limitations set by these dimensional parameters are described in this section.

6.1 Dimensions of the RCG code

Most large or complex atomic structure calculations will fail with the standard RCG code due to dimensional limits specified within the source code at compile time. Generally, in UCD the RCG code is available in four different dimensional sizes. The additional sizes of large (`l`), extra large (`xl`) and extremely large (`xxl`) are described in Table 3 with reference to the size of the dimensional parameters used in the RCG code. For details of these dimensional parameters please consult both the Cowan code manuals and also the RCG source code.

Dimension	normal	large	extra large	extremely large
Parameter		`l`	`xl`	`xxl`
KMX	150	500	990	1500
KJP	150	500	500	500
KTRAN	600	6100	25000	100000
KEXC	40	999	999	999
KPC	2000	2000	12000	12000
KPR	2100	2100	2200	2200
KLSP	420	1500	3000	3000
KLS1	47	119	119	119

Table 3: The values of the RCG dimensional parameters as used in the four different compiled versions of RCG typically used in UCD.

Note also that in the extremely large (`xxl`) version of the RCG code two additional changes were made to the code in that the default values of the internal variables `iengyd` and `nevmax` were adjusted upwards from 500 to 2500 to allow the complete version of a coupling matrix to be printed out when there were more than 500 eigenvalues in any given coupling matrix.

6.2 Limits of the `shrink` code

There are two principal limits on the type of RCG output file that can be processed by the `shrink` code. The first limit is set by the parameter `idim` which by default is set to 1200. This limit is a limit on the number of discrete eigenvalues that there can be described in the whole RCG output file. It is also the maximum number of eigenvalues that there can be for any given total angular momentum. To process RCG output files which have more than this number of eigenvalues for a given total angular momrntum value this dimension will need to be increased.

The second limitation on the `shrink` code is determined by the parameter `ileading` which can have an effect on the memory requirements of the program while executing. This parameter is usually set to 4. This means that for all discrete eigenvalues within the RCG output file, the top four leading eigenvector components will be printed out in the summaries for each eigenvalue. This summary information is stored in the *.sorted* and *.unsorted* output files generated by the `shrink` code. This parameter can be reduced to increase the `idim` parameter where memory is at a premium.

6.3 Limits of `sortbyj`

The `sortbyj` utility is limited in the number of lines that it can sort by the parameter `ilines`. By default this parameter is set to 50000, but may need to be increased if using the utility to sort *.spec* files which have more lines than that.

6.4 Limits of `sortbylower`

Likewise, the `sortbylower` utility is also limited to the number of lines that it can sort by the parameter `ilines`. By default this parameter is set to 20000. Once again, when necessary, this parameter will need to be increased.

6.5 Limits of `splitspec`

The `splitspec` utility is limited to the number of lines that it can read in from a *.spec* file and split into different files. This limitation is set by the parameter `ilines` which by default is set to 20000. When *.spec* files with more transitions than this need to be used then the code will need to be recompiled. There is no limit on the number of unique lower states which may exist among the set of all transitions in the input *.spec* file.

6.6 Limits of `spec2plt`

Three limitations are present in the `spec2plt` program. the first limits the wavelength interval between adjacent points within the internal array used for calculations within the program. This limit is set by the parameter `deltalambda` which by default is set to 0.005Å[3]. The second limititation governed by the parameter `idim` determines the maximum number of points in the calculated plot array. This is normally set to 50001 points and thus the wavelength range of the output plot is normally restricted to a range of 250Å. The third limitation is set by the parameter `ilines` which determines the number of transitions that can be read into an array within the `spec2plt` program. This is usually set to 50000. If the total number of transitions to be included into a single plot using the `spec2plt` program is greater than this, then recompilation of the program using a greater parameter is necessary. Otherwise, there is no limit on the number of *.spec* files that may be combined into one plot.

6.7 Limits of `spectemp`

The maximal number of transitions that the `spectemp` program can process is set by the parameter `ilines` which by default has the value of 20000. The number of distinct lower states which can be processed from any given *.spec* file is limited by the parameter `ilevs` normally set to 1000. If either of these dimensions is insufficent the value of the parameter can be adjusted and the program recompiled.

[3]It was chosen to use a constant wavelength interval instead of a constant energy interval as over a wide spectral range the reciprocal dispersion of most spectroscopic instruments varies only slowly with wavelength.

6.8 Limits of `subgamma`

The only limit in the `subgamma` program is the information on the identification and decay of the upper states of each transition. By default the `subgamma` program can only store information about the decay of 900 upper states. This is determined by the value of the parameter `istates` within the source code. No limit exists on the number of spectral lines in a *.spec* that can be processed as these do not need to be retained in memory.

6.9 Limits of `ewop`

There are no dimensional limits on the `ewop` program as it only needs to store information about one transition contained in the *.spec* file at any one time.

6.10 Limits of `convolve`

At present this section does not apply.

6.11 Limits of `addplots`

The limits which apply to the `addplots` limits are determined by the three parameters `idim`, `ibdim` and `iplots`. The constant wavelength interval is also specified by `deltalambda` and set by default to 0.005Å. The first parameter, `idim`, is a limit on the size of the array required for reading and temporarily storing in memory the output of `spec2plt` which will be combined with one or more other plot files from the same program. This, as in the `spec2plt` program is by default set to 50001.

The second parameter, `ibdim`, limits the final size of the resultant plot. This by default is set to 90000 as the original plot files may not cover exactly the same spectral region. The last parameter `iplots` limits the number of plot files that can be processed by the program. Presently this limit is set to 80 files.

A Examples

A number of small examples are also included in this document. These are examples of small output files, of calculations and running a particular set of calculations in a batch job.