ADAS-EU R(10)PU03

ADAS-EU
ADAS for fusion in Europe

Grant: 224607

Hugh Summers, Adam Foster, Stuart Loch, Martin O'Mullane
and Allan Whiteford

PUBL3: Heavy species in fusion plasma modelling and spectral analysis

25 May 2010

Workpackages : 26-6-3
Category : DRAFT – CONFIDENTIAL

PUBL3: Heavy species in fusion plasma modelling and spectral analysis

Hugh Summers, Adam Foster, Stuart Loch, Martin O'Mullane and Allan Whiteford

Department of Physics, University of Strathclyde, Glasgow, UK

Abstract: *The derived data for usual application in fusion, from the perspective of light elements, are hugely unwieldy for heavy elements, preventing the immediacy and handleability to which ADAS aspires for the experimental diagnostic analyst. So additional work must be done within ADAS in the direction of spectral synthesis for the spectroscopist and in the direction of enhanced condensation for the plasma computational modeller. The purpose of this article is to put all these steps in the hands of the ADAS user who wishes to become expert and, for the more application oriented user of ADAS, to explain what is available now for heavy species in ADAS, how to access it and use it correctly.*

Contents

Preface

This article is the one of a series of technical notes which are being prepared as useful extracts during the longer term construction of the next edition of the ADAS user manual. As such it reflects a change in style, planned for the new manual. It will be more book-like, examining and explaining in detail the physics basis behind the commitment to certain approaches in ADAS and how these work out in practice. The new manual will remain technically detailed with extended appendices. However, it is hoped this will be ameliorated by much more emphasis on worked examples. That is to say the actual manoeuvres, adopted by experienced ADAS users in getting the atomic modelling into application scenarios will be mapped, rather as an expert system. It has become clear that, for some, ADAS operates in a somewhat rarefied atmosphere in which too much is assumed. It is this which I wish to improve upon.

Heavy species do represent a major absorption in the fusion physics community at this time. They are challenging experimentally because of their radiating characteristics and are unfamiliar as plasma contacting wall materials and as migrating plasma species in operating scenarios. Also from a modelling and spectral analysis point of view they are complex with very large numbers of ionisation stages overwhelming models and computers, millions of spectral lines overwhelming identification and simple spectral analysis, and excessive numbers of active bound electrons which stop the most accurate collision cross-section calculations.

In this sense, heavy species have also stopped ADAS for a while. ADAS, stemming from the JET Joint Undertaking, has its roots and greatest sophistication in light elements. The matter has been put right, or writing more cautiously 'improved', with ADAS redesigned. It is hoped that this redesign will appear transparent to many ADAS users. This document is intended for those wish to work more closely with heavy species in the redesigned ADAS framework.

H P Summers
October 2009

Chapter 1

Introduction

The original core purpose of ADAS [4] is to act on collections of fundamental atomic data and to transform them into the useful derived data collections required for direct application to experiment analysis and plasma models. To do so, ADAS implements 'collisional-radiative' atomic models. This purpose remains for heavy species. Unfortunately, the necessary fundamental data collections of items such a energy levels, Einstein coefficients and electron impact excitation rate coefficients are either non-existent or require prohibitive amounts of handwork to assemble. So the redesigned ADAS addresses and enables the task of automatic creation of necessary fundamental data. The derived data for usual application in fusion, from the perspective of light elements, are hugely unwieldy for heavy elements, preventing the immediacy and handleability to which ADAS aspires for the experimental diagnostic analyst. So additional work must be done within ADAS in the direction of spectral synthesis for the spectroscopist and in the direction of enhanced condensation for the plasma computational modeller. The purpose of this article is to put all these steps in the hands of the ADAS user who wishes to become expert and, for the more application oriented user of ADAS, to explain what is available now for heavy species in ADAS, how to access it and use it correctly and to inform on what can reasonably be added on request.

The remainder of this chapter introduces some nomenclature, outlines basic ideas of population and emisssion modelling and encourages the reader, with access to ADAS, to start trying some ADAS procedures. Also a quick pointer is given to where the main ADAS heavy species data are to be found in the data base. Chapter 2 is a 'hands-on' approach to the ADAS tools for calculating a baseline of fundamental data. It follows this through to calculating excited populations, emissivities and radiated power of heavy element ions for the baseline of derived data. Chapter 3, in a similar approach, addresses the calculation of ionisation state with sections on ionisation and recombination coefficient production. In chapter 4, special methods, implemented to ease heavy species handling in plasma transport models, are treated. Chapter 5 describes the ADAS approach to on-going improvement of its heavy element capability, called 'lifting the baseline'. This approach is quite tightly defined and focused and it is hoped that in consequence it will be helpful to external structure/collision specialists, who might wish to contribute to the ADAS data bases. There are a number of appendices for completeness.

1.1 Atomic nomenclatures

A particular state of an ion of a heavy atom has the electrons occupying a set of *orbitals* or *shells*. The specification of the number of electrons in each orbital is called the *configuration*. Thus the ground state of the ion Kr^{+1} has configuration $1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^5$. ADAS incidentally can tell you the ground configuration of any ion of any element, using the ADAS IDL procedure **read_adf00.pro** which accesses data of ADAS data format *adf00*[1]. For the ADAS user, seeking the ground configuration of tantalum, nuclear charge, *z0* = 73 and ion charge[2], *z_ion*=6, type at the IDL command line (user entry is in *italic*)

IDL>*z0=73*

[1]The FORTRAN subroutine **xxdata_00.for** returns the complete data held in ADAS data format *adf00* which includes ionisation potentials.

[2]Conventionally z_0, z and $z_1 = z + 1$ are used for the nuclear charge, the ion charge and the ion charge+1 in formulae and codes. Misleadingly *z*1 has been used as the IDL procedure keyword for the ion charge z. The keyword *z1* is now deprecated in favour of the non-clashing *z_ion*

IDL>*z_ion=6*
IDL>*read_adf00,z0=z0,z_ion=z_ion,config=config*
IDL>*print,config*

giving the result: 1s2 2s2 2p6 3s2 3p6 3d10 4s2 4p6 4d10 4f13 5s2 5p6. The pattern shown is the form adopted by the atomic structure theorist Bob Cowan, whose codes are extensively used for heavy species in ADAS. Note that the orbitals are in the natural order of increasing orbital quantum number l and principal quantum number n. The occupancy of the shell is often assigned the letter q. Cowan may omit closed inner closed shells and also shells of zero occupancy - which can cause uncertainty and confusion. The Cowan configuration string is rather long and more compact forms are used by other atomic structure theorists such as Eissner. These more compact notations are helpful to fit in with the historic organisation of ADAS data formats - especially *adf04*. ADAS adopts two configuration formats, *Standard form* , which for the Ta^{+6} ground state appears as

$1s2\ 2s2\ 2p6\ 3s2\ 3p6\ 3da\ 4s2\ 4p6\ 4da\ 4fd\ 5s2\ 5p6$

Here letters are used for $q > 9$ as $a \equiv 10, b \equiv 11$ and so on, so that the character field length assigned to each orbital substring is 4. Also ADAS uses the *Eissner form* which for the Ta^{+6} ground state appears as

$21522563524565606527568609 63A52B56C$

Here each orbital substring is assigned three characters apart from the first which is assigned only two characters if the shell occupancy is < 10. Thus 21 refers to the $1s^2$ orbital and 609 refers to the $4d^{10}$ orbital. The last character in each set of three is the orbital number as $1 \equiv 1s$, $2 \equiv 2s$, $3 \equiv 2p$, $4 \equiv 3s$, $5 \equiv 3p$, $6 \equiv 3d$, $7 \equiv 4s$, $8 \equiv 4p$, $9 \equiv 4d$, $A \equiv 4f$, $B \equiv 5s$, $C \equiv 5p$ and so on down the alphabet. The first two characters of each orbital substring always give a number equal to $50 + q$. This is except for the first orbital in the string for which the 50 is missed out if $q < 10$. Eissner form is a little confusing at first but very compact. In mass produced *adf04* datasets, the configuration string appears in Eissner form. ADAS has a number of IDL procedures and FORTRAN routines to identify if text strings are of standard form or Eissner form and to transform between them, including **xxdtes.pro** and **xxcftr.pro** in IDL and **xxdtes.for** [3] and **xxcftr.for** in FORTRAN. For example, in IDL type

IDL>*in_cfg = '1s2 2s2 2p6'*
IDL>*type=2*
IDL>*xxcftr,in_cfg=in_cfg,out_cfg=out_cfg,type = type*
IDL>*print,out_cfg*

giving the result 21522563. The procedure actually uses **xxdtes.pro** to check if the input string is correctly formed to be a configuration string and then $type = 1,\ 2,\ 3$ and 4 gives Standard → Standard, Standard → Eissner, Eissner → Standard and Eissner → Eissner respectively.

The words 'resolution', 'resolved' and 'unresolved'are used a lot in ADAS. 'Resolution' in the ADAS context does connect to spectrum lines but in a more abstracted atomic physics sense than in spectroscopy. A configuration describes in general a large number of individual quantum states. The smaller sub-groupings *term* and *level* are more relevant to spectroscopy, identified in Russell-Saunders coupling (LS coupling) by total orbital angular momentum, L, and total spin angular momentum, S, quantum numbers in the first case and by L, S and total angular momentum, J, quantum numbers (or possibly only J if L and S are too imperfect quantum numbers) in the second case respectively. The complete set of spectrum lines between two configurations is called a *transition array*, between two terms is called a *multiplet* and between two levels is called a *component* or simply a *line*. The word 'resolution' refers to whether one is dealing with configurations and transition arrays - referred to as *configuration average* or simply *ca* resolution, with terms and multiplets - referred to as *term* or simply *ls* resolution, or with levels and lines - referred to as *level* or *intermediate coupling* or simply *ic* resolution. Many ADAS data sets are encountered with 'ca_' or 'ls_' or 'ic_' embedded in the data set name and it is to the above resolution that they refer. Resolution or choice of resolution is of great importance in ADAS for heavy species since *ls* or *ic* resolution may easily flood computers and storage. So ADAS must pick carefully amongst the possible resolutions and it is here that the connection with actual spectroscopy occurs. In principle, high resolution spectroscopy allows identification of individual lines, but if there are very large numbers of lines or if the spectral resolution is lower, then only multiplets or possibly only transition arrays can be distinguished. From a fundamental atomic physics point of view, required data on Einstein coefficients and excitation rate coefficients at *ca* resolution are to a first approximation simply weighted sums of these data at *ls* resolution and likewise at *ic* resolution. However, the lower resolutions can be evaluated theoretically much more expeditiously.

[3] A more sophisticated variant of **xxdtes.for** is available called **g5dtes.for** which can split very long configuration strings into 'top' and 'tail' so that the 'top' fits in with older fixed ADAS configuration string lengths. Also the subroutine picks out the valence orbital n and l.

1.2 Population structure

In ADAS, the words *population structure* are used to refer to the set of number densities of excited states of an ion in a thermal plasma relative to the ground state number density at specified conditions of electron density N_e and electron temperature T_e. Usually other parameters, such as deuterium density, deuteron density and deuteron temperature are of less importance. For heavy species modelling, at this time, only the electron parameters are included and so the population structure is determined by electron collisions and by spontaneous emissions alone. Also the free electrons are assumed to be isotropic with a Maxwellian distribution of speeds. In fact ADAS is a bit more particular about 'excited' states, distinguishing *ordinary excited states* which can cascade freely and excited states which cannot. The latter states, which are low lying - that is close to the ground state in energy and which have large slowly relaxing populations relative to ordinary excited states are called *metastable states*. They behave from a time-dependent point-of-view rather like the true ground state and so are grouped with it. Collectively, it is convenient to call the true ground state and the metastables simply the *metastables*. For light elements, the separating off of the metastables from ordinary excited states is important for accurate dynamic modelling of such species in the fusion plasma. This approach is called the *generalised-collisional-radiative picture* or *GCR picture* for short [5]. But what are the metastables in practice? This depends on the ion and its charge and is connected with the 'resolution' discussed in section 1.1. For ions of low charge dominated by the electrostatic part of the Hamiltonian, levels of the same term are nearly degenerate and *ls* resolution is appropriate. The main metastables are mostly then the lowest lying terms of each spin system of the ion - the situation for light element ions. For more highly charged systems, the relativistic part of the Hamiltonian becomes important, spin system breakdown occurs and the levels of terms move apart in energy. Then the main metastables tend to be the fine structure levels of the lowest term and *ic* resolution is appropriate. The *GCR* picture for the low ionisation stages of a heavy element is appropriately at *ls* resolution (which is called *level 1* population modelling[4]) and the high ionisation stages should be at *ic* resolution (which is called *level 2* population modelling[4]). It is however difficult to procure accurate enough atomic data at these resolutions so that meaningful metastable lifetimes may be calculated. *GCR* is too sophisticated for the generality of heavy element ions at this time. So simplification (to a *baseline* approximation of population modelling) is necessary. We do not wish to lose completely the possibility of re-introducing the full *GCR* picture, albeit selectively and consistently, when possible in the future (see chapter 5). So we wish to examine these modelling levels, their interconnections, their assumptions and how to mix them in some detail in the next sub-section 1.2.1. For those who do not wish to follow the matrix algebra of section 1.2.1, it may just be noted that for the heavy species *baseline* modelling, we allow only the ground state to be a 'metastable' and all the rest of the states are treated as ordinary excited states. This latter picture is called the *collisional-radiative* picture or *CR* picture for short. We handle the population structure in a mixture of *ic* and *ca* resolutions which is not prevented by the *CR* picture.

1.2.1 Some algebra

Consider the populations of ionisation stage z, separated into the metastable populations N_ρ^{+z}, indexed by the Greek letter ρ, and ordinary excited populations N_i^{+z}, indexed by the Roman letter i. The stage z has adjacent stages $z-1$ and $z+1$, its *child* and *parent*, with metastable populations labelled as N_μ^{+z-1} and N_ν^{+z+1} respectively. The time-dependent equations 1.1 of the populations are written in matrix/suffix form[5], where we have omitted coupling to more distant ionisation stages.

$$\frac{d}{dt}\begin{bmatrix} N_\mu^{+z-1} \\ N_\rho^{+z} \\ N_i^{+z} \\ N_\nu^{+z+1} \end{bmatrix} = \begin{bmatrix} \mathcal{C}_{\mu\mu'} & N_e\mathcal{R}_{\mu\sigma} & 0 & 0 \\ N_e\mathcal{S}_{\rho\mu'} & C_{\rho\sigma} & C_{\rho j} & N_e r_{\rho\nu'} \\ 0 & C_{i\sigma} & C_{ij} & N_e r_{i\nu'} \\ 0 & N_e S_{\nu\sigma} & N_e S_{\nu j} & \mathcal{C}_{\nu\nu'} \end{bmatrix} \begin{bmatrix} N_{\mu'}^{+z-1} \\ N_\sigma^{+z} \\ N_j^{+z} \\ N_{\nu'}^{+z+1} \end{bmatrix} \tag{1.1}$$

This means that these equations are actually complete only for the stage z. Note that we have not shown explicitly the ordinary populations of stages $z-1$ and $z+1$ and that some of the sub-matrices are shown as script letters (eg. $\mathcal{C}_{\mu\mu'}$ and $\mathcal{R}_{\mu\sigma}$) whereas others are shown as standard letters (eg. $C_{\rho\sigma}$ and $S_{\nu j}$). Technically, this is because a 'quasi-static' assumption has been made about the ordinary populations of the stages $z-1$ and $z+1$ and the influence of their ordinary populations has been condensed onto their metastable populations. Note that the on-diagonal elements of C and $\mathcal{C}$ are -ve quantities. C and $\mathcal{C}$ are linear in the electron density N_e. We wish to demonstrate this procedure for the ordinary populations of the stage z.

[4] *level 1* and *level 2* also imply a precision level for the radiative and collisional data which must be met.
[5] In the following equations summation convention over repeated indices is adopted.

The quasi-static assumption is that $dN_i^{+z}/dt = 0$ which means that these ordinary populations are assumed in instantaneous statistical equilibrium with the various metastable populations[6]. This implies that

$$\begin{bmatrix} N_{\mu'}^{+z-1} \\ N_{\sigma}^{+z} \\ N_{j}^{+z} \\ N_{\nu'}^{+z+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -C_{ji}^{-1}C_{i\rho} & -N_e C_{ji}^{-1} r_{i\nu} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} N_{\mu}^{+z-1} \\ N_{\rho}^{+z} \\ N_{\nu}^{+z+1} \end{bmatrix} \tag{1.2}$$

and then

$$\frac{d}{dt}\begin{bmatrix} N_{\mu}^{+z-1} \\ N_{\rho}^{+z} \\ N_{\nu}^{+z+1} \end{bmatrix} = \begin{bmatrix} \mathcal{C}_{\mu\mu'} & N_e \mathcal{R}_{\mu\sigma} & 0 \\ N_e \mathcal{S}_{\rho\mu'} & \mathcal{C}_{\rho\sigma} & N_e \mathcal{R}_{\rho\nu'} \\ 0 & N_e \mathcal{S}_{\nu\sigma} & \mathcal{C}_{\nu\nu'} \end{bmatrix} \begin{bmatrix} N_{\mu'}^{+z-1} \\ N_{\sigma}^{+z} \\ N_{\nu'}^{+z+1} \end{bmatrix} \tag{1.3}$$

where we have the definitions of the effective metastable cross-coupling coefficient, effective recombination coefficient and effective ionisation coefficients between the various metastables of stages z, $z-1$ and $z+1$

$$\begin{aligned} Q_{\sigma\to\rho}^{cd} &\equiv \mathcal{C}_{\rho\sigma}/N_e &&= (C_{\rho\sigma} - C_{\rho j}C_{ji}^{-1}C_{i\sigma})/N_e \\ A_{\nu'\to\rho}^{cd} &\equiv \mathcal{R}_{\rho\nu'} &&= r_{\rho\nu'} - C_{\rho j}C_{ji}^{-1}r_{i\nu'} \\ S_{\sigma\to\nu}^{cd} &\equiv \mathcal{S}_{\nu\sigma} &&= S_{\nu\sigma} - S_{\nu j}C_{ji}^{-1}C_{i\sigma}. \end{aligned} \tag{1.4}$$

Also there is formally an addition to the $\mathcal{C}_{\nu\nu'}$ term called the parent metastable cross-coupling coefficient

$$X_{\nu'\to\nu}^{cd} \equiv -(S_{\nu j}C_{ji}^{-1}r_{i\nu'})/N_e \tag{1.5}$$

which we assume had already been incorporated. The superscript 'CD' denotes 'collisional-dielectronic' - a historic synonym for 'collisional-radiative' and parallels the naming conventions in the ADAS data format *adf11* used for such data.

The matrix algebra and description above is the most complete formulation within the collisional-radiative framework, but can only be implemented at a consistent *level 1* or *level 2* precision in practice if all the fundamental data ingredients, especially radiative (including forbidden transitions) and collisional (including non-dipole and spin changing) data are available. At the present time, and probably well into the future, this will be possible, and indeed only worthwhile, for selected, targetted ions of special diagnostic impact. In this context though, we do note that rapid progress is being made on the first wave of targetted ions and iso-electronic sequence in the ADAS Project and we return to this in chapter 5. In this section, progressive simplifications of the *GCR* sub-matrix partitions and their evaluation are described which reduce accuracy to *baseline* level, but simultaneously extend coverage to universal. It is helpful at this point to make some distinction between the baseline of fundamental data production and the baseline of derived collisional-radiative data. The fundamental data accuracy imposes limits on the derived data accuracy. Within a baseline accuracy of fundamental data, simplifications of the collisional-radiative methods for production of the derived data are justifiable.

Consider then the two-by-two submatrix partition of the full collisional-radiative matrix (equation 1.1) for the ionisation stage z and the associated one-by two and two-by-one partitions for the ionisation and recombination linkages to the parent ionisation stage $z+1$ respectively. We seek fundamental data for the ionisation stage z (see chapter 2) to fill the matrices $C_{\rho\sigma}$ and of C_{ij}. The ADAS baseline mass production uses the *Cowan* code and its electron impact collisional rates are calculated in a modified plane-wave Born approximation. So spin-changing (except from spin system breakdown) collisional transitions are missing and $C_{\rho\sigma}$ is in principal unsound - as the collisional loss rate from metastables is incomplete. In practice though, metastability decreases for more highly ionised heavy element ions. Also, for heavy species with typically small energy differences between true ground and populated metastables, the metastable-ground collisional coupling in $C_{\rho\sigma}$ would be large and move the relative populations to Boltzmann. We can force the metastable relative populations to Boltzmann or introduce artificial strong collisional coupling rates between them (creating local thermodynamic equilibrium within the metastable set) to the same effect. Let us force Boltzmann fractions of the metastables of the stages $z-1$ and $z+1$. That is

$$\begin{aligned} \left[N_{\mu}^{+z-1}\right] &= \left[f_{\mu}^{Boltz}\right] N^{+z-1} \\ \left[N_{\nu}^{+z+1}\right] &= \left[f_{\nu}^{Boltz}\right] N^{+z-1} \end{aligned} \tag{1.6}$$

[6] We assume no direct populating mechanism from stage $z-1$ to ordinary excited state of stage z.

where $f_{\mu}^{(z-1)Boltz} = \omega_{\mu} exp(I_{\mu}/kT_e)/\sum_{\mu'} \omega_{\mu'} exp(I_{\mu'}/kT_e)$ and $N^{+z-1} = \sum_{\mu} N_{\mu}^{+z-1}$. Since the sum of the populations of the ordinary levels is very small compared with the sum of the metastable populations, N^{+z-1} is the effectively the whole population of the stage $z-1$. Similiarly for $f_{\nu}^{(z+1)Boltz}$ and N^{+z+1}. The I_{μ} denote ionisation potentials. Then adding up equations 1.5 over μ and ν and substituting from equations 1.6

$$\frac{d}{dt}\begin{bmatrix} N^{+z-1} \\ N_{\rho}^{+z} \\ N^{+z+1} \end{bmatrix} = \begin{bmatrix} \mathcal{C}^{(z-1)} & N_e \mathcal{R}_{\sigma}^{(z-1\leftarrow z)} & 0 \\ N_e \mathcal{S}_{\rho}^{(z-1\rightarrow z)} & \mathcal{C}_{\rho\sigma} & N_e \mathcal{R}_{\rho}^{(z\leftarrow z+1)} \\ 0 & N_e \mathcal{S}_{\sigma}^{(z\rightarrow z+1)} & \mathcal{C}^{(z+1)} \end{bmatrix} \begin{bmatrix} N^{+z-1} \\ N_{\sigma}^{+z} \\ N^{+z+1} \end{bmatrix} \tag{1.7}$$

where $\mathcal{C}^{(z-1)} = \sum_{\mu} \mathcal{C}_{\mu\mu'} f_{\mu'}^{Boltz}$, $\mathcal{R}_{\sigma}^{(z-1\leftarrow z)} = \sum_{\mu} \mathcal{R}_{\mu\sigma}$, $\mathcal{S}_{\rho}^{(z-1\rightarrow z)} = \mathcal{S}_{\rho\mu'} f_{\mu'}^{Boltz}$ and so on. Evidently, we can do the same for the metastables of ionisation stage z so returning completely to the *CR* picture as required for our *baseline*. But equations 1.7 demonstrate how to mix *GCR* and *CR* data selectively.

Returning again to the two-by-two submatrix partition of the full collisional-radiative matrix (equation 1.1), we have discussed the strategy for the populations N_{ρ} and the sub-matrix $C_{\rho\sigma}$. From the quasi-static assumption and equations 1.2, the ordinary level populations of the stage z are given by

$$N_j^{+z} = -C_{ji}^{-1} C_{i\sigma} N_{\sigma}^{+z} - C_{ji}^{-1} r_{i\nu} N_e N_{\nu}^{+z+1}. \tag{1.8}$$

It is to be noted that the collisional-couplings in the $C_{i\sigma}$ and C_{ij} are dominated by non-spin change and for these the *Cowan* calculations for the ADAS *baseline* are sound. The ordinary level populations may be expressed therefore as

$$N_j^{+z} = F_{j\sigma}^{(exc)} N_e N_{\sigma}^{+z} + F_{j\nu}^{(rec)} N_e N_{\nu}^{+z+1} \tag{1.9}$$

showing that the ordinary level populations are made up of two parts, driven by excitation from the metastables of the stage z and by recombination from the metastables of the stage $z+1$ respectively, becoming with the ADAS *baseline* simplification

$$N_j^{+z} = F_{j\sigma}^{(exc)} f_{\sigma}^{(z)Boltz} N_e N^{+z} + F_{j\nu}^{(rec)} f_{\nu}^{(z+1)Boltz} N_e N^{+z+1}. \tag{1.10}$$

Sometimes it is preferred to refer the ordinary populations to the ground state (lowest metastable) of the stage. Putting $f_{\sigma 1}^{(z)Boltz} = f_{\sigma}^{(z)Boltz}/f_1^{(z)Boltz}$, then equation 1.10 is replaced by

$$N_j^{+z} = F_{j\sigma}^{(exc)} f_{\sigma 1}^{(z)Boltz} N_e N_1^{+z} + F_{j\nu}^{(rec)} f_{\nu 1}^{(z+1)Boltz} N_e N_1^{+z+1}. \tag{1.11}$$

It is helpful to consider further sub-divisions of the ordinary excited states spanned by C_{ij}. For the ionisation stage z, let $n_{cf}^{(ic)}$ be the number of configurations which we need to include in structure calculations at *ic* resolution for spectroscopy (subject also to the constraint of computational resources). Let $n_{cf}^{(ca)}$ be the number of configurations which we need to include in structure calculations at *ca* resolution to ensure the excitation line power is sufficiently complete. We expect $n_{cf}^{(ca)}$ to include configurations up to some principal quantum n-shell $n_{ns}^{(ca)}$ of the valence electron. Finally let $n_{ns}^{(bn)}$ be number of principal quantum shells of the valence electron which must be considered to ensure dielectronic recombination is sufficiently complete (*bn* stands for 'bundle-n'). Evidently the $n_{cf}^{(ic)}$ should be contained in the set $n_{cf}^{(ca)}$. Also $n_{ns}^{(ca)}$ should be contained in $n_{ns}^{(bn)}$. It is of course unrealistic computationally to evaluate fully-coupled *ic*-populations over all configurations $n_{cf}^{(ca)}$ and to evaluate fully-coupled *ca*-populations over all n-shells $n_{ns}^{(bn)}$. It is equally unrealistic from a physics point-of-view, since collisional processes in finite density plasmas ensure that sub-shell populations of high-enough configurations approach statistical within each configuration and also that *l*-sub-shell populations of high-enough n-shells approach statistical within each n-shell. We have before us then three manageable collisional-radiative matrices, of progressively greater excited level span but of progressively coarser resolution.

$$\begin{bmatrix} C_{ij}^{ic)} \end{bmatrix}, \begin{bmatrix} C_{\bar{i}\,\bar{j}}^{(ca)} & C_{\bar{i}j'}^{(ca)} \\ C_{i'\bar{j}}^{(ca)} & C_{i'j'}^{(ca)} \end{bmatrix}, \begin{bmatrix} C_{\bar{\bar{i}}\,\bar{\bar{j}}}^{(bn)} & C_{\bar{\bar{i}}\,\bar{j}'}^{(bn)} & C_{\bar{\bar{i}}\,j''}^{(bn)} \\ C_{\bar{i}'\bar{\bar{j}}}^{(bn)} & C_{\bar{i}'\bar{j}'}^{(bn)} & C_{\bar{i}'j''}^{(bn)} \\ C_{i''\bar{\bar{j}}}^{(bn)} & C_{i''\bar{j}'}^{(bn)} & C_{i''j''}^{(bn)} \end{bmatrix} \tag{1.12}$$

where partitioning delimits the progressively extending ranges. The notation means that $C_{\bar{i}\,\bar{j}}^{(ca)}$ spans the same range of levels as $C_{ij}^{(ic)}$, but in the former case the components are bundled-up for whole configurations, while in the latter the levels are fully resolved. Again $C_{i'j'}^{(ca)}$ and $C_{\bar{i}'\bar{j}'}^{(bn)}$ span the same range of configurations, but in the latter case are

bundled-up into principal quantum shells. Sophisticated population modelling, as is done in *GCR* modelling of light elements, carries out a sequence of condensations (analogous to that of equations 1.1 → 1.3) and expansions from the three-by-three partition matrix to the two-by-two partition matrix and finally to the one-by-one partition matrix. In illustration, the step from the two-by-two *ca* matrix is as follows: Eliminate the direct couplings in the $C^{(ca)}_{\bar{i}\,\bar{j}}$ partition of the two-by-two matrix and call it $\bar{C}^{(ca)}_{\bar{i}\,\bar{j}}$. Then replace it with $\bar{\bar{C}}^{(ca)}_{\bar{i}\,\bar{j}} = \bar{C}^{(ca)}_{\bar{i}\,\bar{j}} - C^{(ca)}_{\bar{i}j'}(C^{(ca)}_{j'i'})^{-1}C^{(ca)}_{i'\bar{j}}$. Expand the matrix $\bar{\bar{C}}^{(ca)}_{ij}$ over the resolved manifold of $C^{(ic)}_{ij}$ through replacing $C^{(ic)}_{ij}$ by $C^{ic)}_{ij} + W_{i\bar{i}}\,\bar{\bar{C}}^{(ca)}_{\bar{i}\,\bar{j}}U_{\bar{j}\,j}$ The pre-multiplier $W_{i\bar{i}}$ is $\omega_i/\omega_{\bar{i}}$ if $i \in \bar{i}$ otherwise 0. The ωs are statistical weights. The post-multiplier $U_{\bar{j}\,j}$ is 1 if $j \in \bar{j}$ otherwise 0. The current ADAS heavy element *baseline* follows a simpler prescription, however some parts of the more elaborate procedure are on the further development 'road-map'. Hence the detailed description here.

For the ADAS heavy element *baseline* approximate handling of the three *CR/GCR* matrix formulations 1.12, we note some determinants of population structure. Bundle-n modelling extending over effectively the infinite set of excited n-shells (for example as implemented in the representative n-shell method of the ADAS codes ADAS204 and ADAS316) shows that the effective free electron density from the collisional point-of-view is N_e/z_1^7, where $z_1 = z + 1$ for ionisation stage z. So a twenty-five-times ionised ion at a tokamak electron density of $5\text{x}10^{13}$ cm^{-1} behaves like hydrogen in a diffuse nebula, such as Orion, in space. This means that that the critical n-shell, n_{crit} at which collisional excitation competes with spontaneous radiative decay is ~ 1000. For a neutral atom it is ~ 3-4. So possibly excluding the neutral and singly ionised ions, virtually all the main observational spectral emission for heavy species in tokamaks is from excited levels which are freely cascading in n. So only collisional excitation from the metastables is effective in contributing to the populations of excited states below n_{crit} and not upward stepwise excitation via other excited states. Population of an excited n-shell below n_{crit} is followed only by radiative cascade (although there may be collisional re-distribution within the sub-states of an n-shell) through lower n-shells until the metastables are again reached. Excitation to higher n-shells is a strongly decreasing function of n (at least ~ n^{-3}). In the light of the above, available computational resources are such that we can choose $n^{(ca)}_{cf}$ to ensure that the excitation contributions to radiated power (see also section 1.3) are virtually complete. That is evaluated in *ca* approximation using the two-by-two matrix. On the other hand the cut-off $n^{(ic)}_{cf}$ tends to be imposed by computing power restrictions. We would prefer it to be closer to $n^{(ca)}_{cf}$ than we can generally achieve. Let $\mathcal{PLT}^{(ca)}_{n^{(ic)}_{cf}}$ denote the radiated power calculated from a population model at *ca* resolution up to configuration number $n^{(ic)}_{cf}$. This corresponds to the one-by-one matrix of 1.12 but in *ca* not *ic* resolution. Then the ADAS *baseline* radiated power for stage z is

$$\mathcal{PLT} = \mathcal{PLT}^{(ic)}_{n^{(ic)}_{cf}} + \mathcal{PLT}^{(ca)}_{n^{(ca)}_{cf}} - \mathcal{PLT}^{(ca)}_{n^{(ic)}_{cf}} \tag{1.13}$$

So what about the influence of recombination on the excited population structure? The ionisation state, that is the balance of ionisation fractional abundances between stages z and $z + 1$ in actual plasma conditions is in nearly all circumstances such that the direct recombination from the metastables of the stage $z + 1$ into the levels of the set up to $n^{(ca)}_{cf}$ is small compared with the excitation contributions from the metastables of stage z, except into the metastables of z themselves. Radiative recombination at low electron density and typical electron temperatures is almost entirely into the low-lying metastables. The other (and much larger contribution to recombination), that is dielectronic recombination, is principally into highly excited n-shells spanned by the range $n^{(ca)}_{ns}$ up to $n^{(bn)}_{ns}$. The recombination/cascade contribution from these n-shells to the levels up to $n^{(ca)}_{cf}$ is insufficient to overturn the dominance of the contributions from excitation. This means that for the ADAS *baseline* spectrum line emission modelling, the population structure part from $n^{(ca)}_{ns}$ up to $n^{(bn)}_{ns}$ may be omitted. This does not mean that recombination is unimportant - just that it has small direct influence on the principal line emission. It affects it only indirectly through the metastable populations and their ionisation balance. Separated treatment of recombination and ionisation for ionisation state is possible for the ADAS *baseline* and is treated in chapter 3.

1.3 Emissivities

The emission per unit volume per unit time in an observed spectrum line $j \to k$ with upper ordinary excited level population N^{+z}_j may clearly be written as

$$A_{j\to k}N^{+z}_j = \sum_{\sigma} \mathcal{PE}\mathcal{C}^{(exc)}_{\sigma,j\to k}N_eN^{+z}_\sigma + \sum_{\nu} \mathcal{PE}\mathcal{C}^{(rec)}_{\nu,j\to k}N_eN^{+z+1}_\nu \tag{1.14}$$

identifying $\mathcal{PEC}^{(exc)}_{\sigma,j\to k}$,the usual *excitation photon emissivity coefficient*, driven by the metastable σ of the stage z, and $\mathcal{PEC}^{(rec)}_{\nu,j\to k}$, the *recombination photon emissivity coefficient*, driven by the metastable ν of the stage $z+1$. The arguments above for the ADAS *baseline* indicate that the $\mathcal{PEC}^{(rec)}_{\nu,j\to k}$ may be neglected and that we simplify to

$$\mathcal{PEC}^{(exc)}_{j\to k} = \sum_{\sigma} \mathcal{PEC}^{(exc)}_{\sigma,j\to k} f^{(z)Boltz}_{\sigma} \tag{1.15}$$

using the normalisation to the total stage population or, normalising to the ground population (see equation 1.11),

$$\mathcal{PEC}^{(exc)}_{j\to k} = \sum_{\sigma} \mathcal{PEC}^{(exc)}_{\sigma,j\to k} f^{(z)Boltz}_{\sigma 1} \tag{1.16}$$

The total line power arising from excitation from the separate metastable σ of stage z is

$$\mathcal{PLT}^{(exc)}_{\sigma} = \sum_{j,k} \Delta E_{jk} \mathcal{PEC}^{(exc)}_{\sigma,j\to k} \tag{1.17}$$

and from the whole stage is

$$\mathcal{PLT}^{(exc)} = \sum_{\sigma,j,k} \Delta E_{jk} \mathcal{PEC}^{(exc)}_{\sigma,j\to k} f^{(z)Boltz}_{\sigma} \tag{1.18}$$

As discussed in the introduction, there may be very many lines for heavy elements giving a 'grass-like' or 'quasi-continuum' spectrum. Embedded in this may be a number of stronger individualised lines. We wish to work flexibly between the individual line and quasi-continuum viewpoints. Introduce an *envelope feature photon emissivity coefficient*, denoted by $\mathcal{FPEC}$. It is defined on a wavelength interval and is a composite feature arising from very many lines from a single ionisation stage. The $\mathcal{FPEC}$ is suitable as a descriptor in wavelength intervals and at spectral resolutions where the individual component lines are unresolved or only partly resolved. This is the situation with very complex heavy species for which it becomes helpful and economical to handle the envelope feature rather than the individual line emissivity coefficients ($\mathcal{PEC}$s) above. An illustration is given in figure 1.1 for Hf^{+28}. It may helpful in this context to contemplate how nearby lines overlap at typical plasma ion temperatures. ADAS provides a simple IDL procedure **c5dplr.pro** which can be used for this as

```
IDL>ndpix=512
IDL>npix=512
IDL>wvmin=4750
IDL>wvmax=4800
IDL>ndcomp = 2
IDL>ncomp = 2
IDL>wvcomp=[4765,4780]
IDL>emcomp=[1.0d-8,5.0d-9]
IDL>tev = 8000
IDL>amss = 12
IDL>doppler = dblarr(ndpix)
IDL>c5dplr,ndpix,npix,wvmin,wvmax,ndcomp,
    ncomp,wvcomp,emcomp,tev,amss,doppler
IDL>plot,doppler
```

where there are two component lines at wavelengths 4765Å and 4780Å with emissivities 1.0x10^{-8} cm$^3 s^{-1}$ and 5.0x10^{-9} cm$^3 s^{-1}$. The example ion is carbon of atomic mass 12 and ion temperature 8000eV.

1.3.1 Some more algebra

Consider a spectral *region of interest* $[\lambda_0, \lambda_1]$, which may be the range of a particular spectrometer or an interval of special diagnostic value. For configurations I and J, introduce the configuration average energies $E^{(av)}_I$ and $E^{(av)}_J$, the transition array average energy $\Delta E^{(av)}_{IJ} = E^{(av)}_J - E^{(av)}_I$ and wavelength $\lambda^{(av)}_{IJ} = hc/|E^{(av)}_J - E^{(av)}_I|$. Configurations such that the transition wavelength $\lambda^{(av)}_{IJ} \in [\lambda_0, \lambda_1]$ should be handled at high resolution, that is *level resolved* (see below), while configurations such that the transition wavelength $\lambda^{(av)}_{IJ} \notin [\lambda_0, \lambda_1]$ may be handled at low resolution, that is in

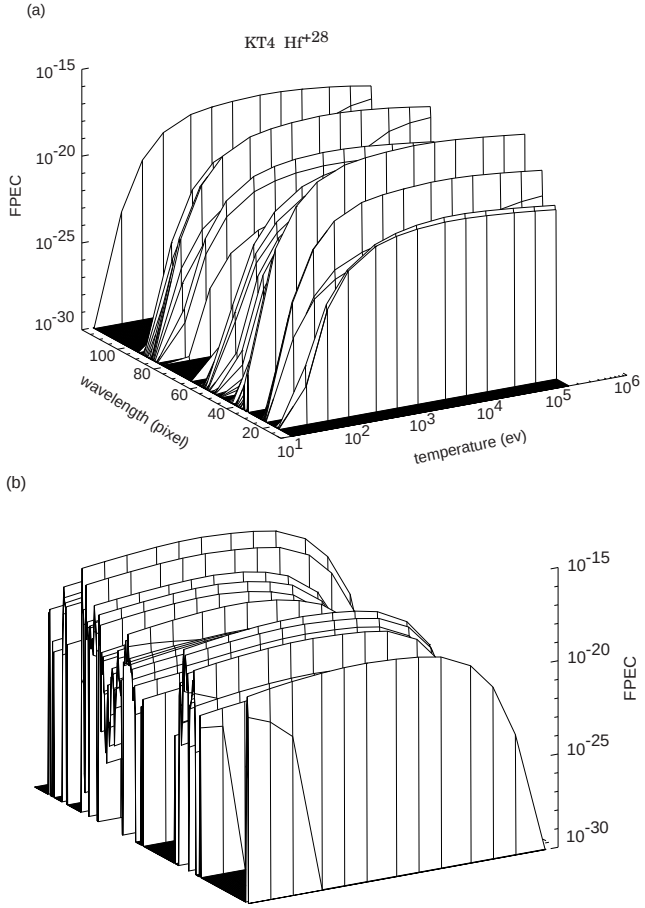

Figure 1.1: $\mathcal{FPEC}$ for Hf^{+28} in the spectral range of the XUV ('SOXMOS' - KT4) spectrometer at the JET facility Vs electron temperature at the fixed electron density of $5.0 \times 10^{+13}$ cm^{-3}. The wavelength scale is detector pixel space. (a) View from the low temperature side. (b) View from the high temperature side showing the broadening into an envelope feature.

configuration average, in which the whole configuration is treated as one effective energy level.

Consider again the spectral interval $[\lambda_0, \lambda_1]$, written compactly as [0, 1] and subdivided into N_{pix} intervals as

$$\begin{aligned}\{\Delta\lambda_i^{[0,1]} \quad &\equiv \quad [\lambda_0 + i(\lambda_1 - \lambda_0)/N_{pix}, \lambda_0 + (i+1)(\lambda_1 - \lambda_0)/N_{pix}] : \\ &\quad\; i = 0, \cdots, N_{pix} - 1\}\end{aligned} \tag{1.19}$$

Also suppose that the $j \rightarrow k$ spectrum line has a normalised emission profile $\phi_{j\rightarrow k}(\lambda)$. In general, such a profile is a convolution of Doppler and instrumental functions. Then the envelope feature photon emissivity coefficient vector is defined as

$$\mathcal{FPEC}_{\sigma,i}^{(exc)[0,1]} = \sum_{j,k:\lambda_{j\rightarrow k}\in[0,1]} \mathcal{PEC}_{\sigma,j\rightarrow k}^{(exc)} \int_{\lambda_i}^{lambda_{i+1}} \phi_{j\rightarrow k}(\lambda) d\lambda \tag{1.20}$$

$\lambda_{j\rightarrow k}$ is the natural wavelength of the $j \rightarrow k$ spectrum line. The default broadening assumed is Doppler with a Maxwellian distribution for the emitting ion at temperature, T_{ion}, equal to the electron temperature, T_e, used in the collisional-radiative modelling of the $\mathcal{PEC}$s. This consitutes a minimum broadening. The integral in the equation is then expressible in terms of error functions as

$$\begin{aligned}\mathcal{FPEC}_{\sigma,i}^{(exc)[0,1]} \quad &= \quad \sum_{j,k:\lambda_{j\rightarrow k}\in[0,1]} \mathcal{PEC}_{\sigma,j\rightarrow k}^{(exc)} \frac{1}{2}\{erfc((\lambda_i - \lambda_{j\rightarrow k})/\sigma) - \\ &\qquad\qquad erfc((\lambda_{i+1} - \lambda_{j\rightarrow k})/\sigma)\}\end{aligned} \tag{1.21}$$

where $\sigma = \lambda_{j\rightarrow k}\alpha\{(kT_{ion}/I_H)(m_p/m_X)(m_e/m_p)\}^{1/2}$ and m_X is the emitting ion mass. α is the fine structure constant. In actual spectral analysis, although the emission of a heavy ion in a spectral interval may be most conveniently handled as an $\mathcal{FPEC}$, there may be a few prominent lines of the ion in the spectral interval which are usefully separated from the $\mathcal{FPEC}$. In practice, when executing the computational modelling of a heavy ion population structure, the $\sim$ 50 strongest line emissivities are ranked and for these the familiar $\mathcal{PEC}$s are archived as well as the $\mathcal{FPEC}$. In principle, the minimal Doppler broadened $\mathcal{FPEC}$s can be convolved with effective instrument functions and/or representations of wavelength dependent filters - see the ADAS series 4 codes ADAS414 and ADAS415. As for the $\mathcal{PEC}$s, the ADAS *baseline* metastable simplification gives $\mathcal{FPEC}_{i}^{(exc)[0,1]}$ and $\mathcal{FPEC}_{1,i}^{(exc)[0,1]}$.

Finally, it can be useful to obtain just the total radiated line power in the spectral interval [0, 1] say of an $\mathcal{FPEC}$.

1.4 Primary ADAS data for heavy species

The minimum atomic data required to calculate the excited population structure of an ion in the CR picture comprises the element symbol, ion charge, nuclear charge and ionisation potential, list of energy states together with their configuration and other quantum number designations, average orbital binding energies and then Einstein A-values and Maxwell averaged electron impact collision strengths for all relevant transitions between pairs of states. These data assemblies are archived in ADAS data format *adf04*. ADAS has *baseline* data of this form for many elements and has semi-automatic codes for their preparation . The data are archived in subdirectories

/home/<uid>/adas/adf04/copmm#<elem. symb.>/.

In general there are three data set types, namely,

ca#<elem. symb.><ion charge>.dat
ls#<elem. symb.><ion charge>.dat
ic#<elem. symb.><ion charge>.dat

corresponding to the *ca*, *ls* and *ic* approximations described earlier.

$\mathcal{PEC}$ and $\mathcal{FPEC}$ coefficients are functions of both electron density, N_e, and electron temperature, T_e. $\mathcal{PEC}$s are archived in ADAS data format *adf15* and the $\mathcal{FPEC}$s in ADAS data format *adf40*. The mass produced heavy element $\mathcal{PEC}$ data have been assigned the year number '06'. Also the *baseline* $\mathcal{PEC}$ data for each ion, referred to the ionisation stage (or ground state only) population, is called the '01' *partition* (*Partitions* are the subject of chapter 4). The *adf15* data is archived in data sets and subdirectories such as

/home/<uid>/adas/adf15/pec40#w/pec40#w_ca#w6.dat

for the ion W^{+6} of tungsten of nuclear charge 74, and so on for *ls* and *ic* types. The associated $\mathcal{FPEC}$s are archived as

/home/<uid>/adas/adf40/fpec40#w/fpec40#w_ca#w6.dat

Finally there is the radiated power data. All the partial $\mathcal{PLT}$ for the ionisation stages of an element are gathered together in one data set. This is archived in subdirectories of ADAS data format *adf11* as for example

/home/<uid>/adas/adf11/plt40_ca#w.dat

ADAS provides IDL procedures to access these ADAS data formats and deliver the results over temperature and density vectors of your choice. Thus **read_adf15.pro** reads *adf15*. The additional IDL function **adas_vector.pro** is helpful for setting up equally-spaced temperature and density vectors (logarithmic by default, or linear). For example, type at the IDL command line

```
IDL>te=adas_vector(low=1.0e1,high=1.0e3,num=10)
IDL>dens=adas_vector(low=1.0e13,high=1.0e14,num=11,/linear)
IDL>block=2
IDL>file='/home/<uid>/adas/adf15/pec40#w/pec40#w_ca#w6.dat'
IDL>read_adf15, file=file,block=block,te=te,dens=dens,wlngth=wlngth,data=data
IDL>print,'wlngth=',wlngth
IDL>print,'te(3)=',te(3),'dens(4)=',dens(4)
IDL>print,'pec(3,4)=',pec(3,4)
```

Chapter 2

Complex atom calculations

For the heavy element baseline of concern here, *ab initio* fundamental atomic data are generated. The first step is the choice of a suitable basis of states for each ion of each element which spans the requirements for population modelling and spectral prediction. It is usual and helpful to develop such a basis of states by focusing on configurations commencing with the configuration to which the ground state of the ion belongs. Further excited configurations are added following certain *promotional rules*, that is prescriptions for moving one (or possibly more than one) electron from one shell of the ground configuration to another (usually higher) shell. Thus for the ground state of krypton a typical promotion is

$$1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 \rightarrow 1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^5 4d^1 \tag{2.1}$$

This promotion is the most likely to occur in an electron impact excitation reaction with the ground state. Radiative transitions from the excited configuration will be main contributors to spectral line emission and net radiated power by neutral krypton in a plasma.

2.1 Promotional rules

Distinguish open and closed shells of the ground configuration. For example, the complete set of core shells up to $4f^{14}$ of neutral tungsten, W^0, has configuration

$$1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 4d^{10} 4f^{14} \tag{2.2}$$

and the outer part of the W^0 configuration is

$$5s^2 5p^6 5d^4 5f^0 5g^0 6s^1 \tag{2.3}$$

showing $5s$ and $5p$ *closed* shells, two *open* active valence shells $5d$ and $6s$ and two embedded *empty* shells $5f$ and $5g$. Promotions of the active valence electrons in the open shells are the first to consider.

The set of promotions for a valence electron from a specific open shell $n_1 l_1$ are specified by the change permitted in n_1 and l_1, that is the set of promotions to shells nl with

$$\begin{aligned} min\_dn\_v1 \leq n - n_1 \leq max\_dn\_v1 \\ min\_dl\_v1 \leq l - l_1 \leq max\_dl\_v1. \end{aligned} \tag{2.4}$$

The rule for the $n_1 l_1$ valence electron is then the values *min_dn_v*1, *max_dn_v*1, *min_dl_v*1 and *max_dl_v*1 and for the $n_2 l_2$ valence electron is the values *min_dn_v*2, *max_dn_v*2, *min_dl_v*2 and *max_dl_v*2. In these ranges, note that *min_dn_v*1, *min_dl_v*1, *min_dn_v*2 and *min_dl_v*1 can be negative. The ADAS procedures distinguish at most two open valence electron shells and allows separate specification of the range maxima and minima in n and l for each. Also, if there is only one valence shell, that is one actual open shell, then the outermost closed shell may be treated as a second

valence shell as though it were open. Likewise if the two outermost shells are in fact closed, as for Kr^0 with ground configuration $4s^2 4p^6$, then the $4p$ shell be may handled as the first valence shell and the $4s$ as the second valence shell.

Next, the promotions from closed shells are considered. A range of closed shells, from which promotion is allowed to occur, that is closed shells $n_{cl}l_{cl}$ are specified, in the range

$$\begin{aligned} min\_n\_cl \leq n - n_c \leq max\_n\_cl \\ min\_l\_cl \leq l - l_c \leq max\_l\_cl. \end{aligned} \tag{2.5}$$

For these closed shells, ranges of permitted change (the same ranges apply to all the designated closed shells) are specified as

$$\begin{aligned} min\_dn\_cl \leq n - n_{cl} \leq max\_dn\_cl \\ min\_dl\_cl \leq l - l_{cl} \leq max\_dl\_cl. \end{aligned} \tag{2.6}$$

The rule is then the values *min_n_cl*, *max_n_cl*, *min_l_cl*, *max_l_cl*, *min_dn_cl max_dn_cl*, *min_dl_cl* and *max_dl_cl*.

It is convenient to add some additional controls and promotional possibilities as follow:

prom_cl	promote from inner shell closed shells (on/off)
*fill_n_v*1	add all *nl* configurations of outer valence shell *n* (on/off)
fill_par	if *fill_n* add only opposite parity else add both parities (on/off)
*last_*4*f*	shift one valence electron to unfilled $4f$ as an extra ground (on/off)
grd_cmplx	include configs. of same complex as the ground config. (on/off)

Note that the rules are not exclusive. In practice, they are worked through successively, adding with each rule only these configurations which have not already occurred.

The elements up to radon ($z_0 = 86$) and all their ions comprise 180 distinct ground configurations. The excess over 86 is because neutral and near neutral ions of a given iso-electronic sequence often have different ground configurations from the more highly ionised members. A typical variation of effective ground configuration is the set
$1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 4d^{10} 4f^{14}$
$1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 4d^{10} 4f^{13} 5s^1$
$1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 4d^{10} 4f^{12} 5s^2$
from the neodymium iso-electronic sequence. ADAS holds promotional rules for every possible ground configuration in ADAS data format *adf54*. The procedure **read_adf54.pro** reads in a complete *adf54* data set as

IDL>*a54file = '/home/<uid>/adas/adf54/promotion_rules_large.dat'*
IDL>*read_adf54,file=a54file,fulldata=ref_rules*
IDL>*print,ref_rules.config[0]*
IDL>*print,ref_rules.min_l_cl[0]*

ref_rules is an IDL structure containing the promotional rules for every possible ground state (hence the vectors are of length 180) . It is defined as:

index[]	:	index of ground configuration of each ion of element in *adf54* file
config[]	:	ground configuration for each ion of element
n_el[]	:	number of electrons for each ion of element
no_v_shl[]	:	number of open (valence) shells. Include outer-most shell even if closed.
max_dn_v1[]	:	maximum Δn promotion for first (outer-most) valence shell.
min_dn_v1[]	:	minimum Δn promotion for first (outer-most) valence shell. Negative value allows access to inner unoccupied or open shells
max_dl_v1[]	:	maximum delta Δl promotion for first (outer-most) valence shell.
min_dl_v1[]	:	minimum delta Δl promotion for first (outer-most) valence shell.
max_dn_v2[]	:	maximum Δn promotion for second (inner-most) valence shell.
min_dn_v2[]	:	maximum Δn promotion for second (inner-most) valence shell.
max_dl_v2[]	:	maximum delta Δl promotion for second (inner-most) valence shell.
min_dl_v2[]	:	minimum delta Δl promotion for second (inner-most) valence shell.
prom_cl[]	:	promote from inner shell closed shells (1=yes,0=no).
max_n_cl[]	:	maximum inner shell n from which promotions are permitted.
min_n_cl[]	:	minimum inner shell n from which promotions are permitted.
max_l_cl[]	:	maximum inner shell l from which promotions are permitted.
min_l_cl[]	:	minimum inner shell l from which promotions are permitted.
max_dn_cl[]	:	maximum Δn promotion from a permitted inner shell.
min_dn_cl[]	:	minimum Δn promotion from a permitted inner shell. Negative values of Δn allow access to inner unoccupied or open shells.
max_dl_cl[]	:	maximum Δl promotion from a permitted inner shell.
min_dl_cl[]	:	minimum Δl promotion from a permitted inner shell.
fill_n_v1[]	:	add all nl configurations of outer valence shell n (1=yes,0=no).
fill_par[]	:	if *n_fill* only add opposite parity to valence shell else add both parities (1=yes, 0=n0).
for_tr_sel[]	:	Cowan option for radiative transitions 1 - first parity, 2 or 3(default).
last_4f[]	:	shift an electron valence shell to unfilled 4f as extra ground.
grd_cmplx[]	:	include configurations of same complex as ground configuation for valence n-shell.

ADAS does have a capability to 'size' proposed complex atom calculations to available computer resources, providing an *adf54* optimised for these resources. This is discussed in section 2.4. In practice it can be useful to obtain the ground configurations and promotional rules for every ion of an element at the one time. The IDL procedure

adas8xx_promotion_rules.pro

achieves this. For example, for krypton ($z_0 = 36$), type at the IDL command line

```
IDL>  z0_nuc=36
IDL>  a54file =                          '/home/adas/adas/adf54/promotion_rules_large.dat'
IDL>  adas8xx_promotion_rules,           z0_nuc = z0_nuc,
                                         a54file = a54file,
                                         ionpot = ionpot,
                                         prom_rules = prom_rules
IDL>  print,'ground config. for z=2 :',  prom_rules.config[2]
IDL>  print,'max_dn_v1 value for z= 2 :', prom_rules.max_dn_v1[2]
IDL>  print,'index value for z= 2 :',    prom_rules.index[2]
```

The structure *prom_rules* has the same organisation as *ref_rules* but the range of the vectors is $0 \rightarrow z_0 - 1$. The index number of the ground state of each ion of the element in the full *ref_rules* list is given by *prom_rules.index*. By adding the keyword *z_ion*, it is possible to specify the exact ionisation stage(s) for which rules are required.

The further procedure

adas8xx_promotions.pro

establishes the actual configurations and their shell occupancies, for an ion of the element. For example continuing from the previous IDL, type at the IDL command line

```
IDL>  z_ion=2
IDL>  adas8xx_promotions,   z0_nuc = z0_nuc,
                            z_ion = z_ion,
                            ionpot = ionpot[z_ion],
                            prom_rules = prom_rules,
                            promotion_results=promotion_results

IDL>  print, promotion_results.no_configs,format='("no_configs =",7i8)'
IDL>  print, promotion_results.no_terms, format='("no_terms =",7i8)'
IDL>  print, promotion_results.no_levels, format='("no_levels =",7i8)'
IDL>  print, 'total configs =',long(total(promotion_results.no_configs))
IDL>  print, 'total terms =',long(total(promotion_results.no_terms))
IDL>  print, 'total levels =',long(total(promotion_results.no_levels))
```

promotion_results is another IDL structure specifying all the configurations of an ion allowed by the *rules* in ground and excited configuration categories, along with their parities and effective charges (of the type required for Cowan code input), together with an occupation number vector expansion of these configurations [7]. *configs* is defined as follows:

grd_cfg	:	ground configuration string (Cowan)
ex_cfg[]	:	excited configuration strings (Cowan)
grd_par	:	ground configuration parity
ex_par[]	:	excited configuration parities
grd_zc	:	grd. configuration eff. charge for Cowan *adf34* driver
ex_zc[]	:	exc. configuration eff. charges for Cowan *adf34* driver
oc_store[,]	:	configuration occupation number vectors (all)

The various output keyword sub-parameters of *configs* are in the necessary form for the subsequent calculations described in the next sub-section. In particular, the ground configuration character string *grd_cfg* and the vector of excited configuration character strings *ex_cfg*, together with their parities, *grd_par* and *ex_par*, are required for determining the atomic structure for the ion. The full details of the output parameters are given in the header lines of the procedures themselves (see appendix B). For complex, many-electron ions, the rules can lead to a substantial number of included configurations and to very large numbers of terms and levels. It is helpful, and part of the machinery of ADAS, to be able to assess these numbers before embarking on the complete structure calculations themselves. As can be seen from the example above **adas8xx_promotions.pro** also gives this information. In illustration, the summary information from the procedure, using the default 'large' promotion rules for W^{+21} gives configurations

$4d^{10}4f^{7}$	ground config.
$4d^{10}4f^{6}5s^{1}$	valence $4f$ shell promotion
$4d^{10}4f^{6}5p^{1}$	
$4d^{10}4f^{6}5d^{1}$	
$4d^{10}4f^{6}5f^{1}$	
$4d^{10}4f^{6}5g^{1}$	
$3d^{10}4s^{2}4p^{5}4d^{10}4f^{8}$	closed $4p$ shell promotion
$3d^{10}4s^{2}4p^{5}4d^{10}4f^{7}5s1$	
$3d^{10}4s^{2}4p^{5}4d^{10}4f^{7}5p1$	
$3d^{10}4s^{2}4p^{5}4d^{10}4f^{7}5d1$	
$3d^{10}4s^{2}4p^{5}4d^{10}4f^{7}5f1$	
$3d^{10}4s^{2}4p^{5}4d^{10}4f^{7}5g1$	

[7] An IDL procedure **cfg2occ.pro** converts from Eissner, Cowan or standard configuration strings to occupation number vector form.

$4d^9 4f^8$	closed $4d$ shell promotion
$4d^9 4f^7 5s^1$	
$4d^9 4f^7 5p^1$	
$4d^9 4f^7 5d^1$	
$4d^9 4f^7 5f^1$	
$4d^9 4f^7 5g^1$	

The configuration, term and level counts separated into those from the sequence of rules are

		g	v1	v2	cl	fill_n	last_4f	grd_cmplx
no_configs	=	1	5	0	12	0	0	0
no_terms	=	31	1186	0	18468	0	0	0
no_levels	=	327	12989	0	231385	0	0	0

with total configs = 18, total terms = 19685 and total levels = 244701. Clearly, at *ls* and *ic* resolution, complete structure calculations may barely be feasible on the largest computers. At *ca* resolution, on the other hand, the structure calculations are feasible on small systems.

2.2 Structure, populations and emissivities

Following the determination of a suitable set of configurations for an ion, there are a number of atomic structure and collisional cross-section codes which can be used to calculate the energy levels, spontaneous emission coefficients and electron impact excitation rate coefficients, and then assemble them into *adf04* data sets in the various resolutions. Such codes will be reviewed and their role for ADAS discussed further in chapter 5. In ADAS, very extensive use is made of the *Autostructure* code and the *Cowan* code. These are both essentially structure codes, and both are able to calculate plane wave Born approximation electron impact cross-sections and rate coefficients. This rather simple approximation (with some threshold modification) is nonetheless very suitable for the ADAS *baseline* general survey of heavy species. *Autostructure* is preferred for dielectronic recombination coefficient calculations and as a prelude to sophisticated *R-matrix* collision calculations. ADAS series 7 is largely centred on use and exploitation of *Autostructure* and we shall discuss it further in section 3.2. For the heavy element *baseline* though, the preferred code is *Cowan* and ADAS series 8 is largely centred on its use and exploitation. It is for this reason that the promotional IDL procedures have the prefix **adas8xx_** since they reside in the series 8 libraries.

Basic execution of *Cowan* is allowed interactively from the IDL-ADAS menus via ADAS801. The input screen asks the user to select a driver data set of format *adf34*. ADAS makes considerable use of driver data sets so that calculations can be exactly reproduced at later times and these drivers are assigned their own ADAS data format numbers. The *adf34* format is designed for *Cowan* code input and it includes the character strings of the ground and excited configurations of the ion obtained in the previous section. Some samples are present in the ADAS data base, archived in element name sub-directories. Thus a driver for a Xe^{+10} *Cowan* structure calculation is **/home/adas/adas/adf34/xenon/xe10.dat**. ADAS can help in preparing *adf34* drivers with the procedure

adas8xx_create_drivers.pro

It requires the configuration set information in the IDL structure *configs* from a prior execution of procedure **adas8xx_promotions.pro**. Illustrative IDL commands are as follow (continuing the IDL for the same earlier Kr^{+2} case, where the keyword parameters *z0_nuc*, *z_ion*, *ionpot* etc. were set):

```
IDL>  files =                    {adf34_file: 'adf34_krypton_kr2.dat'}
IDL>  adas8xx_create_drivers,    z0_nuc = z0_nuc,
                                 z_ion = z_ion,
                                 ionpot = ionpot[z_ion],
                                 files = files,
                                 promotion_results=promotion_results
```

Note that a restricted keyword parameter set has been used in this illustration. Return to the example of an **adas8xx_create_drivers.pro**

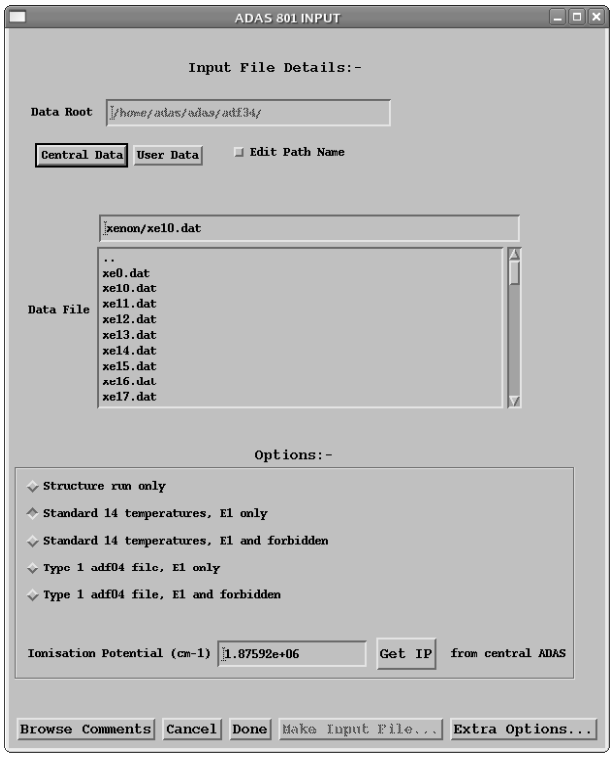

Figure 2.1:

call shown above. The *adf34* data set is ready for input to ADAS801 in an interactive IDL-ADAS sesssion. The input screen for ADAS801 is shown in figure 2.1. Note the expectation of an *adf34* driver, which can be from the user ADAS file space. ADAS801 returns a fully formed *adf04* datasets in both *ls* and ic approximation ready for further ADAS processing.

Structure and Born cross-section calculation in the *ca* approximation are simpler than than for the *ls* and *ic* cases. Only the *rcn1* step for single electron orbital energies and *rcn2* step for uncoupled one-electron matrix elements are required from the *Cowan* package. The simplicity means that it is quick to execute even for ions with extended configuration lists and so can provide approximate estimates of the contributions of configurations which are precluded by size in the much more complex *ls* and *ic* calculations. An IDL procedure has been prepared in a 'stand-alone' form for generation of *adf04* data sets at *ca* resolution, namely:

adas8xx_create_ca_adf04.pro

The procedure requires an occupancy matrix for included configurations, data which generally comes in the *promotion_results.oc_store* keyword sub-parameter from previous execution of **adas8xx_promotions.pro**. The ionisation potential and a set of temperatures for tabulation of the Maxwell averaged collision strengths are required for the *adf04* file. Again, a restricted set of keyword parameters have been used. It is necessary to generate a set of reduced temperatures, theta. A simple direct call for C^{+3} with configurations $1s^2 2s^1$, $1s^2 3d^1$, $1s^2 3s^1$, $1s^2 3p^1$ and $1s^2 2p^1$ is illustrated below.

```
IDL>  z0_nuc = 6
IDL>  z_ion=3
IDL>  adf04_t1_file =          'adf04_copmm#6_ca#c3_t1.dat'
IDL>  adf04_t3_file =          'adf04_copmm#6_ca#c3_t3.dat'
IDL>  ionpot=64.494

IDL>  occup=                   [ [2,1,0,0,0,0],
                               [2,0,0,0,0,1],
                               [2,0,0,1,0,0],
                               [2,0,0,0,1,0],
                               [2,0,1,0,0,0]
                               ]
IDL>  theta=                   [ 2.00e+02, 3.00e+02, 5.00e+02, 7.00e+02, 1.00e+03, $
                               1.50e+03, 2.00e+03, 3.00e+03, 5.00e+03, 7.00e+03, $
                               1.00e+04, 1.50e+04, 2.00e+04, 3.00e+04, 5.00e+04, $
                               7.00e+04, 1.00e+05, 1.50e+05, 2.00e+05, 3.00e+05, $
                               5.00e+05, 1.00e+06, 2.00e+06, 5.00e+06, 1.00e+07]
IDL>  plasma=                  { theta: theta,$
                               theta_noscale:[0],$
                               indx_theta:indgen(n_elements(theta)) }
IDL>  adas8xx_create_ca_adf04, z_ion,
                               z0_nuc,
                               occup,
                               ionpot = ionpot,
                               plasma = plasma,
                               adf04_t1_file = adf04_t1_file,
                               adf04_t3_file = adf04_t3_file
```

The output *adf04* files, *adf04_t1_file* and *adf04_t3_file*) are of type 1 (collision strength) and of type 3 (Maxwell averaged collision strength) respectively and are fully formed for further use in ADAS calculations. Note the use of the *plasma* structure here: this structure is designed to hold details of the plasma parameters and spectrometer wavelength ranges. Here, we have only defined three items within the structure; the full structure is defined as follows:

theta[]	:	electron temperature vector(K)
indx_theta[]	:	index vector sub-selection from full *theta* vector
theta_noscale	:	1 (or set)=> temperatures *theta* are not scaled with z_1
		0 (or not set) => temperatures *theta* are scaled with z_1
rho[]	:	electron density vector(cm^{-3})
indx_rho[]	:	index vector sub-selection from full *rho* vector
rho_scale	:	1 (or set)=> densities *rho* are scaled with z_1
		0 (or not set) => densities *rho* are not scaled with z_1
npix[]	:	wavelength ranges pixel allocation vector
wvlmin[]	:	minimum wavelength (Å) of each wavelength range
wvlmax[]	:	maximum wavelength (Å) of each wavelength range
indx_wvl[]	:	index vector sub-selection from full set of wavelengths ranges

It is convenient to work with z-scaled temperatures and densities in collisional-radiative modelling. Conventionally one defines $\theta = T_e/z_1^2$ and $\rho = N_e/z_1^7$ where $z_1 = z + 1$. ADAS has for many years used a tabulation of θ which is approximately equally spaced logarithmically. Computational limitations in earlier years meant that the ranges used were less than we now prefer. For compatibility and flexibility, the ADAS default θ set is quite wide and dense and we sub-select from it with a pointer vector called *indx_theta*. The older ADAS *indx_theta* pointed to eight *theta* values, but now fourteen values are usually selected. A logical *unscaled_theta* advises if the working *theta* vector is not scaled. The default, with the keyword omitted is scaled. A similar policy is adopted for ρ, although here the default is for the density to be unscaled. Since we have a number of preferred spectral intervals, matching the various spectrometers at JET and at the laboratories of ADAS Project participants, an *indx_wvl* pointer approach is also adopted for the wavelength intervals. Note that *npix* gives the detector pixel allocations for the various intervals. A default *plasma* structure can be loaded by calling the function *adas8xx_plasma_defaults.pro*:

IDL> *plasma = adas8xx_plasma_defaults(/all)*

The population structure, radiated power and emssivity calculation for an ion may be executed interactively by selecting the code ADAS810 from the ADAS series 8 menu. The input screen for ADAS810 is shown in figure 2.2. Note that there is an option of using a newly created *adf04* data set (type 3 is required) or an *adf42* data set. Format

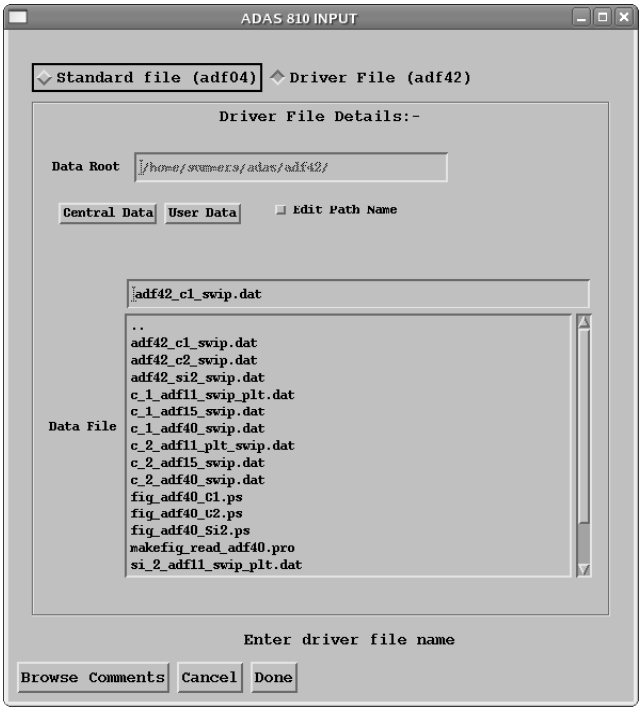

Figure 2.2:

adf42 is assigned to drivers for ADAS810. Such drivers are convenient since, as well as specifying the *adf04* file to use, they specify all the additional parameters required for execution (such as electron temperatures and densities), so that the subsequent ADAS810 processing screen is filled. Also the names of the various output data sets are specified, all this assisting systematic studies. As perhaps expected, the IDL procedure **adas8xx_create_drivers.pro** can also create the *adf42* driver. Returning to the Kr^{+2} example, using the sample *promotion_results* structure:

```
IDL>  plasma =                adas8xx_plasma_defaults(/all)
IDL>  files =                 { adf34_file : 'adf34_ca_kr2.dat' , $
                              adf42_ca_file : 'adf42_ca_kr2.dat', $
                              adf04_ca_file : 'adf04_ca_kr2.dat', $
                              adf40_ca_file : 'adf40_ca_kr2.dat', $
                              adf15_ca_file : 'adf15_ca_kr2.dat', $
                              adf11_ca_file : 'adf11_ca_kr2.dat'}
IDL>  adas8xx_create_drivers, z0_nuc = z0_nuc, $
                              z_ion = z_ion, $
                              ionpot = ionpot[z_ion], $
                              files = files, $
                              promotion_results=promotion_results, $
                              plasma = plasma
```

With the restricted set of keyword parameters in this example, the various output data set names (*adf11, adf15* and *adf40*) are set up to to match the *adf42* name by the *adf42* driver.Note that the filenames here are not of the form favoured for archiving, and are simply kept short as an example. The *files* structure will be explained in more detail in section 2.3.

2.3 Automatic running and naming conventions

We wish to move on to large scale production and now describe the full capabilities built in to the ADAS IDL codes. It is useful for this to be able to define some structure to move large numbers of variables around simply as structures. Already the plasma structure has been defined in section 2.2, holding the plasma conditions and spectrometer properties in which the ion is being studied. Also of use when dealing with large quantities of data is a standardised naming convention, not only for final data sets but also for the many different driver files required to operate the codes.

The ADAS driver data sets of format *adf34* and *adf42* have been introduced in the previous sub-section. Quite a large number of data sets are created, used and archived in the heavy species work and we wish to maintain consistency and logical connection of naming for our heavy species production. Another IDL structure, *files* is introduced, defined as follows:

adf34_file	:	primary (config.) *adf34* driver name for ADAS801
adf34_inst_file	:	supple. (plasma) *adf34* driver name for offline ADAS8#1
adf34_ls_pp_file	:	supple. (*ls*-resol.) *adf34* driver name for offline ADAS8#1
adf34_ic_pp_file		supple. (*ic*-resol.) *adf34* driver name for offline ADAS8#1
adf42_ca_file	:	primary (*ca*-resol.) *adf42* driver name for ADAS810
adf42_ls_file	:	primary (*ls*-resol.) *adf42* driver name for ADAS810
adf42_ic_file	:	primary (*ic*-resol.) *adf42* driver name for ADAS810
adf42_ca_pp_file	:	supple. (*ca*-resol.) *adf42* driver name for offline ADAS810
adf42_ls_pp_file		supple. (*ls*-resol.) *adf42* driver name for offline ADAS810
adf42_ic_pp_file	:	supple. (*ic*-resol.) *adf42* driver name for offline ADAS810
adf04_ca_t1_file	:	specific ion (*ca*-resol.) *adf04* type 1 (Ω) dataset name
adf04_ca_t3_file	:	specific ion (*ca*-resol.) *adf04* type 3 (Υ) dataset name
adf04_ls_file	:	specific ion (*ls*-resol.) *adf04* type 3 (Υ) dataset name
adf04_ic_file	:	specific ion (*ic*-resol.) *adf04* type 3 (Υ) dataset name
adf15_ca_file	:	$\mathcal{PEC}$ emiss. coefft. (*ca*-resol.) *adf15* dataset name
adf15_ls_file	:	$\mathcal{PEC}$ emiss. coefft. (*ls*-resol.) *adf15* dataset name
adf15_ic_file	:	$\mathcal{PEC}$ emiss. coefft. (*ic*-resol.) *adf15* dataset name
adf40_ca_file	:	$\mathcal{FPEC}$ feature emiss. coefft. (*ca*-resol.) *adf40* dataset name
adf40_ls_file	:	$\mathcal{FPEC}$ feature emiss. coefft. (*ls*-resol.) *adf40* dataset name
adf40_ic_file	:	$\mathcal{FPEC}$ feature emiss. coefft. (*ic*-resol.) *adf40* dataset name
adf11_ca_file	:	$\mathcal{PLT}$ power coefft. (*ca*-resol.) partial-*adf11* dataset name
adf11_ls_file	:	$\mathcal{PLT}$ power coefft. (*ls*-resol.) partial-*adf11* dataset name
adf11_ic_file	:	$\mathcal{PLT}$ power coefft. (*ic*-resol.) partial-*adf11* dataset name

Note the various names for final derived data output (*adf11*, *adf15* and *adf40*), the fundamental data *adf04* and the drivers *adf34* and *adf42* in the *ca_*, *ls_* and *ic_* resolutions. Also note the extra *inst_* and *pp_* names within *adf34* and *adf42* which are required for batch processing.

Structure and modified Born collisional rate ceofficient calculation in the *ls* and *ic* approximation can be performed at the IDL command line by the procedure

adas8xx_create_ls_ic_adf04.pro

This procedure requires the full capabilities of ADAS8#1 and is designed to fit in with use of *adf34* drivers and the systematic naming conventions of the heavy element baseline. It requires the supplementary *adf34* drivers of type *_inst* and *_pp* as well as the main driver. Suppose these are present in the data base for Fe^{+22} (perhaps through prior use of **adas8xx_create_drivers.pro**), then in illustration, at the IDL command line enter

```
IDL>  z0_nuc = 26
IDL>  z_ion=22
IDL>  adf34_file =          /home/adas/adas/adf34/iron/fe22.dat
IDL>  adf34_inst_file =     /home/adas/adas/adf34/iron/fe22_inst.dat
IDL>  adf34_ls_pp_file =    /home/adas/adas/adf34/iron/fe22_ls_pp.dat
IDL>  adf34_ic_pp_file =    /home/adas/adas/adf34/iron/fe22_ic_pp.dat
IDL>  adf04_ls_file =       strcompress('adf04_copmm#' + string(z0_nuc, $
                            format='(i2)') + '_' + 'ls#' + elsymb $
                            + string(z_ion],format='(i2)') + '.dat', $
                            /remove_all)
IDL>  adf04_ic_file =       strcompress('adf04_copmm#' + string(z0_nuc, $
                            format='(i2)') + '_' + 'ic#' + elsymb $
                            + string(z_ion],format='(i2)') + '.dat', $
                            /remove_all)
IDL>  files =               { z0_nuc = z0_nuc , $
                            z_ion = z_ion, $
                            files = files, $ }
```

In this vein, the calculation may be continued at the IDL command line with the next stage of population structure and emissivity evaluation, delivering *adf11*, *adf15* and *adf40* output data sets. As noted earlier, the ADAS810 package contains the key codes for these calculations with drivers *adf42*. Again, assuming setup of the *files* structure in a prior step (so that the *adf42* drivers are created correctly), enter

```
IDL>  adas8xx_create_adf15_adf40,   z0_nuc = z0_nuc ,
                                    z_ion = z_ion,
                                    files = files
```

The latter calculation can be restricted to only the *ca* part with the keyword */ca_only*, otherwise the three *ca*, *ls* and *ic* calculations will be executed.

To alleviate the need to run all of these procedures manually, the IDL **run_adas808.pro**.
can be used as a wrapper for this. There are a number of keyword variable parameters and default settings so that input to the procedure can be small. An example at the IDL command line is as follows

```
IDL>  z0_list = [25,26]
IDL>  nel_min = 3
IDL>  nel_max = 4
IDL>  a54file = '/home/adas/adas/adf54/promotion_rules_medium.dat'

IDL>  run_adas808,   z0_list=z0_list, nel_min=nel_min, $
                     nel_max=nel_max,/ca_only,/only_801
or

IDL>  run_adas808,   z0_list=z0_list, nel_min=nel_min, $
                     nel_max=nel_max, a54file=a54file, /only_801
or

IDL>  run_adas808,   z0_list=z0_list, nel_min=nel_min, $
                     nel_max=nel_max,a54file=a54file
```

A list of elements is specified at the one time as, for example *z0_list=[10,12]*, and all the ions of these elements with numbers of bound electrons between *nel_min* and *nel_max* are computed at the same time. The keyword *only_ca*

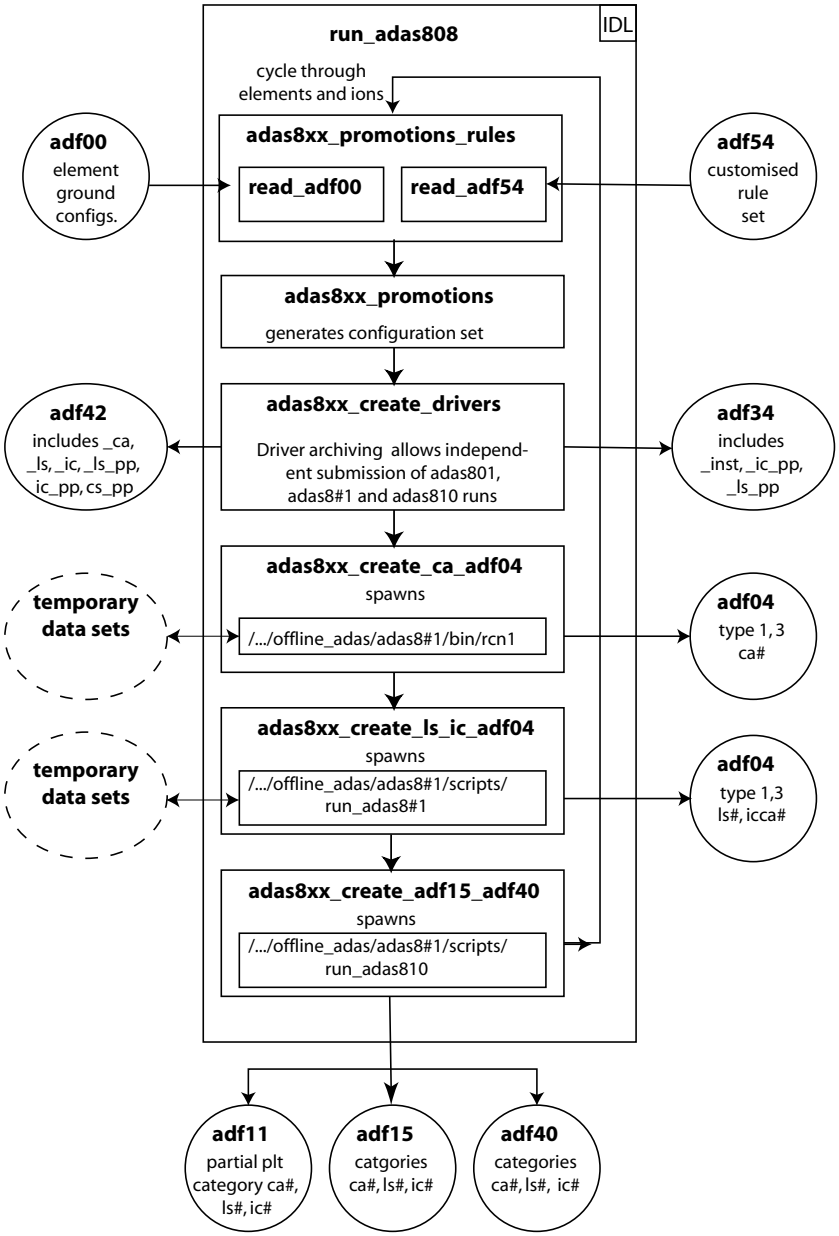

Figure 2.3: Schematic of **run_adas808.pro** program flow for complete sequential heavy species calculation from the IDL command line. The various temporary and permanent data sets created are indicated. Note that the **adas8xx_create_ls_ic_adf04.pro** and **adas8xx_create_adf15_adf40.pro** steps spawn **offline_adas** scripts.

restricts the calculation to configuration average only. This is advisable for a first quick look at a new heavy element. The keyword *only_801* terminates the calculation at the atomic structure point, that is after creation of the *adf04* files. Otherwise the calculation continues on to the population structure, low level line power and emissivities. The procedure by default creates the output data sets in the directory from which the procedure is executed. However the keyword */archive_files* will cause creation of sub-directories following conventional ADAS data format naming prescriptions. These will again be in the current directory, unless the keyword *archive_dir* is set to specify an alternative location. In fact scripts for off-line execution of ADAS801 and ADAS810 are also created during the execution of **run_adas808.pro** with the terminating part-name *_script*, together with information text files with the terminating part-name *_paper*.

adf no.	subdir.	subsubdir.	datasets
adf04/	copmm#<nuc.chge.> /		ca_40#<el.symb.><ion chge.>.dat
			ca_40#<el.symb.><ion chge.>_t1.dat
			ls_40#<el.symb.><ion chge.>.dat
			ic_40#<el.symb.><ion chge.>.dat
adf11/	<class>40		<class>40_<el.symb.>.dat
adf11/*	plt40_partial	plt40#_partial_<el.symb>	plt40_partial_ca#<el.symb.><ion chge.>.dat
			plt40_partial_ls<el.symb.><ion chge.>.dat
			plt40_partial_ic#<el.symb.><ion chge.>.dat
adf15/	pec40#<el.symb.>/		pec40#<el.symb>_ca#<el.symb><ion chge.>.dat
			pec40#<el.symb>_ls#<el.symb><ion chge.>.dat
			pec40#<el.symb>_ic# <el.symb><ion chge.>.dat
adf40/	fpec40#<el.symb> /		fpec40#<el.symb>_ca#<el.symb><ion chge.>.dat
			fpec40#<el.symb>_ca#<el.symb><ion chge.>.dat
			fpec40#<el.symb>_ca#<el.symb><ion chge.>.dat
adf34/	<elem.name> /		<el.symb.><ion chge.>.dat
			<el.symb.><ion chge.>_inst.dat
			<el.symb.><ion chge.>_ls_pp.dat
			<el.symb.><ion chge.>_ic_pp.dat
adf42/	<elem.name> /		ca#<el.symb.><ion chge.>.dat
			ls#<el.symb.><ion chge.>.dat
			ic#<el.symb.><ion chge.>.dat
			<el.symb.><ion chge.>_ca_pp.dat
			<el.symb.><ion chge.>_ls_pp.dat
			<el.symb.><ion chge.>_ic_pp.dat
scripts/	<elem.name> /		<el.symb.><ion chge.>_ls_801_script
			<el.symb.><ion chge.>_ic_801_script
			<el.symb.><ion chge.>_ca_810_script
			<el.symb.><ion chge.>_ls_810_script
			<el.symb.><ion chge.>_ic_810_script

Table 2.1: 2. The classes under ADAS data format *adf11* comprise *acd*, *ccd*, *scd*, *qcd*, *xcd*, *plt*, *prb*, *prc*, *zcd*, *ycd*, *ecd*. The last three are required for superstaging and are new to ADAS. 3. * denotes a temporary archive of partial *plt* data which is conveniently held in local storage. It does not appear in central ADAS. 4. '810' refers to population, low level line power and emissivity post-processing and is described in the next section.

A little more needs to be said about directories and sub-directories. ADAS has a significant legacy of sub-directory structures and data set naming practice within the various ADAS data formats. Some of these have become so entrenched that they have assumed the status of 'the standard naming' although they have some inconsistencies between different ADAS data formats and with hindsight might have been chosen differently. While not a major issue for work with light elements, it is so for heavy elements driven by the need to operate automatically. In this section, the preferred prescriptions for heavy elements are given and comprise a somewhat eclectic mixture recognizing historical usage but sometimes the need to change. Year number, element nuclear charge, element name, ion charge, partition layer and legacy mnemonic conventions play a part. The naming conventions, by ADAS data format, are shown in table 2.3. Note that to cope with the advent of these heavy species calculations, the 'year' number conventions have been

updated, with all work derived from this heavy species baseline being allocated the number 40. Further improvements later will be allocated sequentially higher numbers (41,42 etc). Superstaged data (see chapter **??**) will be allocated year numbers starting from 60 upwards.

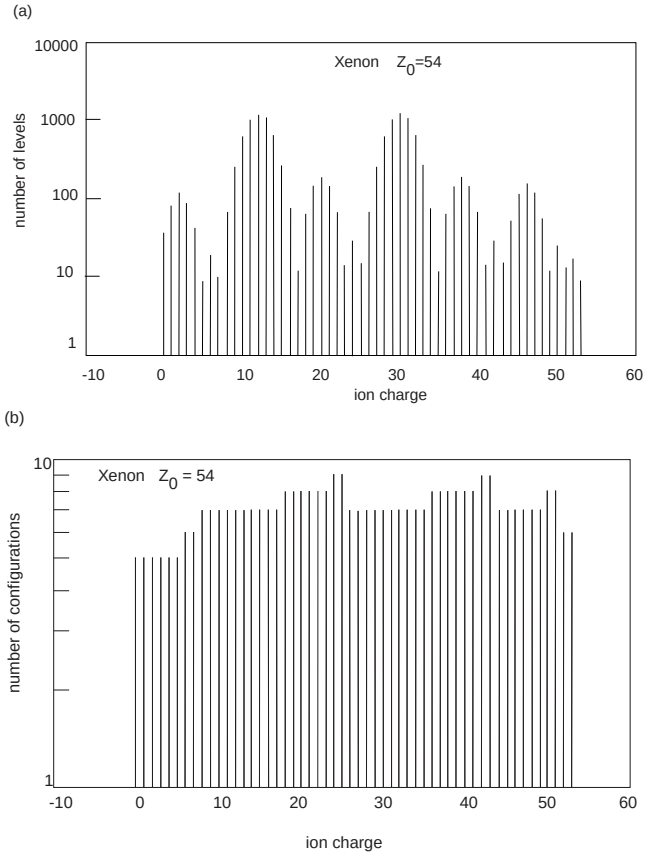

Figure 2.4: Projected structure calculation size for Xenon. The strategy was single electron promotions with $\Delta n \leq 2$, $l' \leq 2$, for valence shells including promotion to inner empty shells. Three spectral regions of interest were delimited spanning the usual XUV, VUV and visible spectrometer ranges. (a) Total number of levels included for each ionisation stage. (b) Number of configurations included for each ionisation stage.

As the **run_adas808.pro** procedure executes, it provides lists of the configurations adopted and the configuration, term and level count estimates for each ion, in the *_paper* file. These are produced even with the */only_ca* keyword set and the information is useful for assessing the overall size of the computations versus computer power and resources. Figure 2.4 has been assembled from such information.

In practice, command line execution of **run_adas808.pro** with its sequential processing of elements and ions of each element is most suited to initial surveys with moderate configuration sets and preliminary studies or exploration of selected cases. Large scale calculations of all the ions of many heavy elements with large configuration sets and level lists are naturally suited to distributed background processing.

run_adas808.sh

is a shell script to implement background processing. It is stored in the */home/adas/offline_adas/adas8#4* directory It is currently enabled for the JAC computers at the EFDA-JET Facility with *LOADLEVELLER* as the job submission and management system, although it can also be called in standalone mode, or made to stop after generating the run scripts to allow use of a different load balancing scheme. A typical execution command for the hydrogen-like to lithium-like ionisation stages of magnesium is

> run_adas808.sh mg 12 magnesium 1 3 /home/<user>/work
/home/<user>/adas/adf54/promotion_rules_mg_adf54.dat

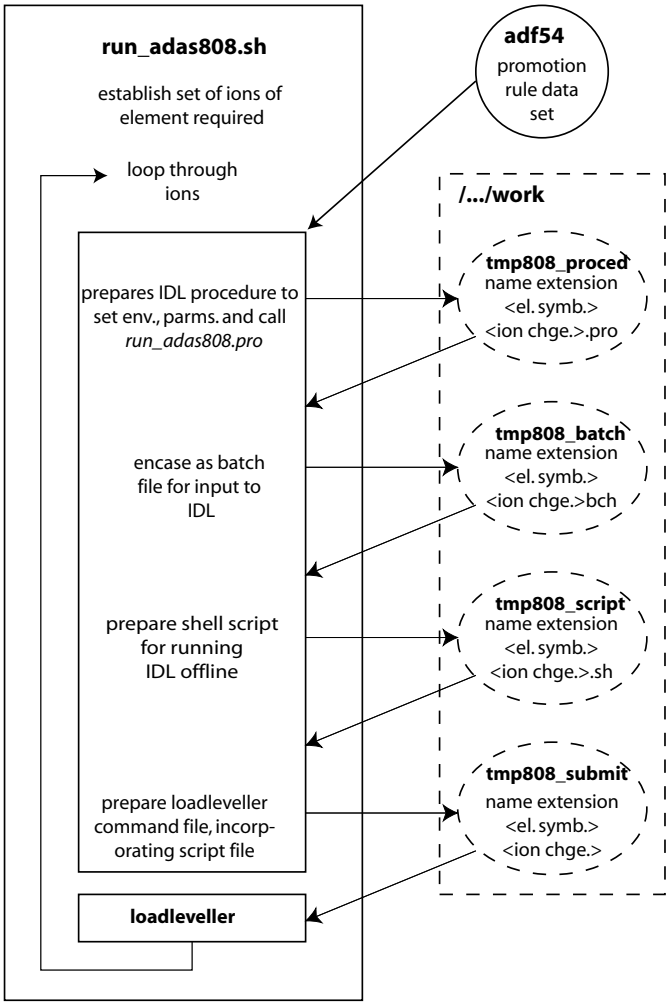

Figure 2.5: Data sets, temporary files and principal loops in **run_808_offline.sh**

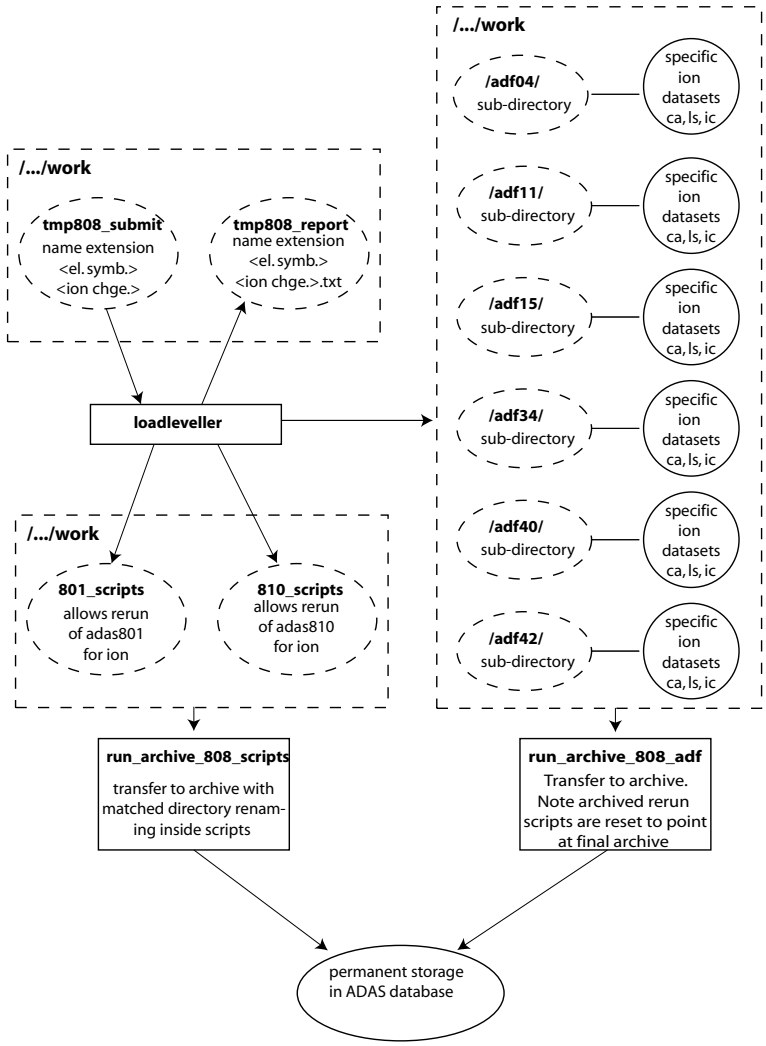

Figure 2.6: Organisation and disposition of data sets used and created in distributed off-line execution managed by LOADLEVELLER

where the ordered parameters are

1.	elsymb	:	element symbol (lowercase)
2.	z0_nuc	:	nuclear charge of element
2.	elname	:	element name (lowercase)
3.	nel_min	:	number of electrons on first ion of element to be calculated.
4.	nel_max	:	number of electrons on last ion of element to be calculated.
5.	work_dir	:	working directory (full pathway); usually */home/<user>/work*.
6.	a54file	:	adf54 file (full pathway).

in addition there are other optional arguments. The position of these arguments is irrelevant:

-q	–quiet	:	run quietly, surpressing paper file generation
-l	–loadleveller	:	run using loadleveller (JET specific)
-d	–donotrun	:	stop after generating the inputs for 801/810 (useful for submitting jobs in non-JET systems)
-y	–year <number>	:	year identifier for generated data (default = 40)
-c	–caonly	:	run only *ca* calculation, not full *ca, ic, ls* set
-s	–scale <number*5>	:	custom Cowan slater parameter multipliers in % (eg: 95 94 90 85 95). Defaults usually sufficient.
-i	–idl <command>	:	custom idl command if required, default is ”idl”

If the loadleveller flag is set, the script has to prepare a number of temporary files leading finally to the *LOADLEVELLER submit* file. These are shown in the schematic of figure 2.5. Finally figure 2.6 shows the organisation and disposition of the various data sets associated with the distributed execution under LOADEVELLER. If the loadleveller flag is not set, the same temporary files are generated (excluding the loadleveller submit file), and the code is launched. This can take some time (several hours per stage × many stages) therefore the -d flag may be used to generate all the temporary files without launching them. The resulting codes can then be run on difference machines using whatever load balancing method you have available.

In the example above, note the choice of *adf54* data set from central ADAS. The ADAS database contains predefined *adf54* data sets for different elements and aimed at different sizes of computer. These may be helpful for the ADAS user working on a laptop, moderate workstation or with larger distributed computational support, based on experience at the EFDA-JET Facility. If a different data set is required, it is advised that the size optimising described in the next section is utilised to obtain a good promotion rule set.

2.4 Optimised sizing for computer systems

It is immediately apparent in preparing *adf54* data sets that computational limitations have to be taken into account. For many stages it is easy to prescribe promotion rules which overwhelm computer resources, whereas for many others the natural promotion rules produce small calculations, which can easily accommodate greater numbers of configurations. Note that with the Cowan code the size of the computation rises almost linearly with the number of levels (*ls* and *ic* resolution) or the number of configurations (*ca* resolution). For most purposes at the EFDA-JET Facility, a calculation with approximately 1000 levels present will enable the calculation to run in a few hours and is usually large enough to encompass the major radiating transitions. There are however certain situations where larger calculations are necessary, in particular those ions with open 4f shells, where even a minimal calculation with only the ground and one excited configuration will have over 3000 levels. The ideal structure/cross-section computation will capture as much of the radiated power as possible in a reasonably sized calculation. It is of course insufficient merely to truncate at such reasonably sized calculations. The radiated power deficit in the truncated calculation must be estimated and the spectral regions where excluded transition arrays lie should be identified. In the ADAS heavy species provision, we seek to answer these problems by playing between *ic*, *ls* and *ca* resolutions, as described later in this section.

To enable custom limits to be placed on calculations, another set of routines have been designed for optimising the databases of rules used in ADAS808. Originally it was hoped that the predefined rule sets could be used to hold this data in its entirety, however it turns out that there are often differences in the optimum promotion rules between different ions even when they share the same ground configuration. The reason for this is twofold: the computation will automatically reject certain combinations of configurations which cause instability, which may differ between isoelectronic sequence members; and the strongest radiating transitions can change from element to element along an isoelectronic sequence. The principle on which the optimisation code operates is fairly straightforward. Since the objective of the optimisation is to include the main radiating transitions, and the majority of the impurity emission is line radiation from collisional excitation/spontaneous emission, the figure of merit has been chosen to be the ratio $\mathcal{PLT}/\Delta n_{levels}$, that is the increase in total line power vs increase in number of levels.

Given a starting configuration, a reference T_e and N_e, and the element and stage of interest, a minimal set of

ID	Rule	Change	ID	Rule	Change
1	$\delta n_{v1_{max}}$	+1	13	ground complex	1
2	$\delta n_{v1_{min}}$	-1	14	fill $v1$ n shell	1
3	$\delta l_{v1_{max}}$	+1		fill same parity only	1
4	$\delta l_{v1_{min}}$	-1	15	fill $v2$ n shell	1
5	$\delta n_{v2_{max}}$	+1		fill same parity only	0
6	$\delta n_{v2_{min}}$	-1	16	extra $4f$ ground	-1
7	$\delta l_{v2_{max}}$	+1	17	$\delta n_{v1_{max}}$	+1
8	$\delta l_{v2_{min}}$	-1		$\delta l_{v1_{max}}$	+1
9	$\delta n_{cl_{max}}$	+1	18	$\delta n_{v2_{max}}$	+1
10	$\delta n_{cl_{min}}$	-1		$\delta l_{v2_{max}}$	+1
11	$\delta l_{cl_{max}}$	+1	19	$\delta n_{c1_{max}}$	+1
12	$\delta l_{cl_{min}}$	-1		$\delta l_{c1_{max}}$	+1

Table 2.2: The changes to the promotion rules attempted in each interation of **adas8xx_opt_promotions_control.pro**.

promotion rules is used. The code progresses iteratively, trying each of 16 different rule changes (listed in table 2.2). For each rule which produces a new configuration set, the Cowan code is run in *ca* only mode (this mode is chosen due to its rapid run time, measured in seconds, while *ic* and *ls* runs can take days with complex configuration sets), and post-processed to obtain the $\mathcal{PLT}$. Assuming this run is successful, the $\mathcal{PLT}/\Delta n_{levels}$ ratio is calculated and stored. After all 16 changes have been attempted, the change which produced the largest ratio is chosen as the reference case, and the process is repeated, using this data set as the initial conditions for the next 16 changes. This continues until the number of levels reaches the target number chosen at the beginning. An example of the step by step nature of this process is shown in figure 2.7,while the structure of the code is shown in figure 2.8. Once finished the code writes the data to an *adf54* file, allowing an ADAS808 run to be performed as before.

The optimisation calculation is executed by **run_optimise_promotion_rules.sh**, in the */home/adas/offline_adas/adas8#4*

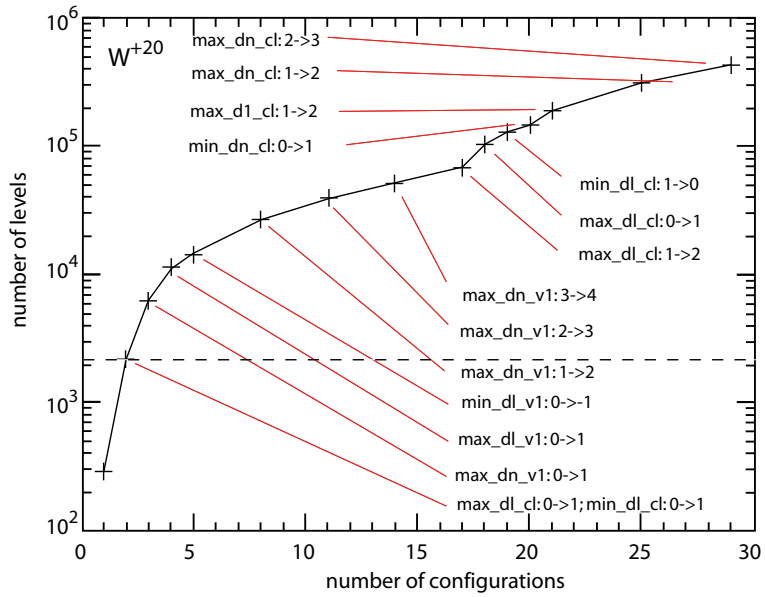

Figure 2.7: The step by step nature of an **adas8xx_opt_promotions_control** run, for W^{+20} (ground configuration: $4d^{10}4f^{8}$). Starting from the ground configuration, promotion rules are amended gradually adding to the total number of configurations. The dotted line indicates where a reasonably large *ic* or *ls* coupled calculation would have stopped.

directory. It a distributed task under LOADLEVELLER, initiated as

>run_optimise_promotion_rules.sh mg 12 magnesium 1 3 /home/<user>/work 1000
/home/<user>/adas/adf54/promotion_rules_magnesium_adf54.dat -l -v

where the ordered parameters are

1. el_symb : element symbol, lowercase.
2. z0_nuc : nuclear charge of element (0 causes default z0_nuc set to be used).
3. el_name : element name, lower case.
4. nel_min : number of electrons on first ion of element to be calculated.
5. nel_max : number of electrons on last ion of element to be calculated.
6. work_dir : working directory (full pathway); usually */home/<user>/work*.
7. size_tgt : target calculation size for computer system.
8. a54file : adf54 file (full pathway).

in addition there are other optional arguments. The position of these arguments is irrelevant:

-v	–verbose	:	Generate paper file with diagnostic information
-l	–loadleveller	:	run using loadleveller (JET specific)
-t	–terms	:	set to optimise in terms of number of terms, not levels (in this case input #7 is the number of terms, not levels)
-y	–year <number>	:	year identifier for generated data (default = 40)
-i	–idl <command>	:	custom idl command if required, default is “idl”

Again, the -l flag is provided for submitting each different stage to another machine on the JAC cluster. If not set, the code will launch each stage sequentially. For heavy elements (e.g. tungsten) this procedure should complete within about a day.

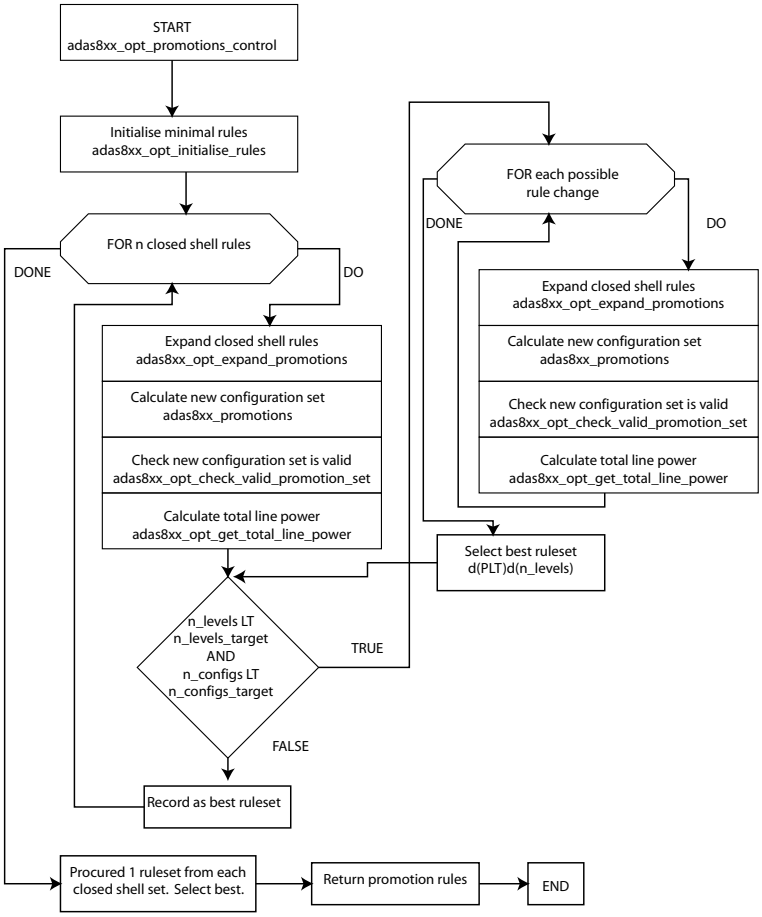

Figure 2.8: The structure of the promotion rule optimisation code. The promotion rules applied during each iteration of **adas8xx_opt_expand_promotions** are listed in table 2.2. In the first iteration, an electric dipole transition is forced to occur to initiate the process

To complete this section, some studies with tungsten are illustrated using the above codes. All results are obtained from the standard output $\mathcal{PLT}$, $\mathcal{PEC}$ and $\mathcal{FPEC}$ data sets in the various resolutions, together with information from the *summary* and *paper* data sets which log the execution of **run_opt_808_offline.sh** and **run_808_offline.sh**. It is hoped that the user will find evident the benefits of production of useful data without extensive atomic knowledge, increased chance of a stable outcome due to error checks on the Cowan code during the promotion rule selection and the ability to manage the calculation size. To explore the physics, calculations for tungsten have been performed in two limits. The first set was limited to 1000 levels, or 30 configurations, whichever limit was encountered first (called the *normal* run). The second set was limited only to the 30 configurations, with an infinite number of levels allowed (called the *large* run). The relative sizes of these calculations are shown in figure 2.9. Note that these are sizing calculations which may be executed in *ca_only* mode.

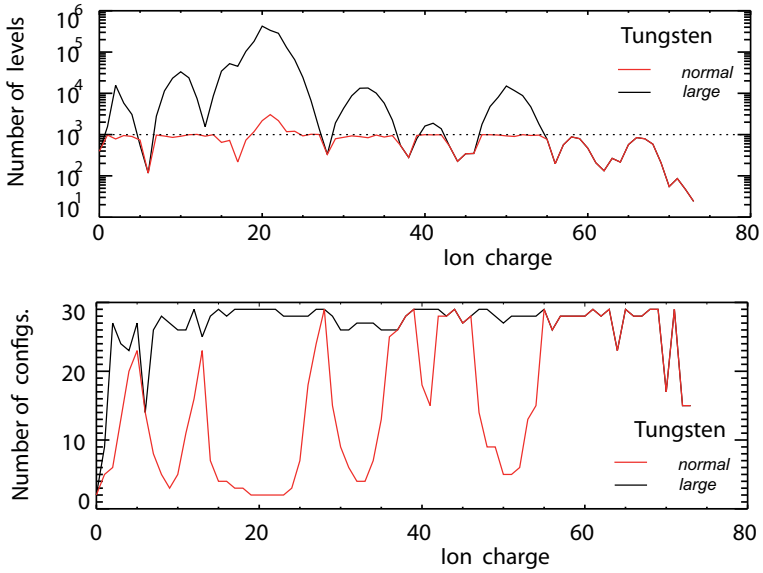

Figure 2.9: The number of levels expected and configurations present in the data sets produced by the *normal* and *large* runs of ADAS808. The dashed line in the upper graph indicates the target calculations size for the *normal* sized run, 1000 levels. The *normal* sized run is executed to completeness in *ic* and *ls* approximation.

Both the integrated emission and the spectral emission of radiated light from individual stages are relevant. In the former case, comparing the $\mathcal{PLT}$ calculations for different resolution is informative, shown in figure 2.10 for two ionisation stages of tungsten. The $\mathcal{PLT}$ for different resolutions can vary quite widely, particularly in stages with open *d* and *f* shells (about 14 stages of tungsten are affected). These discrepancies always follow the pattern of *ic* being the most heavily reduced, then *ls*, then *ca*. The differences are attributed to the energy of the ground state in each model being different. The energy of the ground configuration of the *ca* model is effectively the weighted mean of the levels of the *ic* model for the same configuration. This leads to systematic differences in the radiated power for systems where there is a large distribution of energies in the ground configuration. The *ic* calculation is the most accurate, and is assumed to be the true radiated power for the included configurations. The difference between the *large* and *normal* sized *ca* runs can be added to this to obtain a good estimate of the full radiated power.

It is therefore ideal to be able to produce a $\mathcal{PLT}$ for each element based upon this $\mathcal{PLT}_{ic} + \mathcal{PLT}_{ca\_large} - \mathcal{PLT}_{ca}$ combination. To aid production of these, a variety of IDL codes and scripts has been produced to produce the completed file. A sample set of commands to produce this could be:

> run_optimise_promotion_rules.sh mg 12 magnesium 1 12
/home/<user>/work/<normdir> 1000
/home/<user>/adas/adf54/promotion_rules_mg_adf54.dat -l -v

to generate the “normal” sized run, and then for the “large” run:

> run_optimise_promotion_rules.sh mg 12 magnesium 1 12
/home/<user>/work/<lrgedir> 999999
/home/<user>/adas/adf54/promotion_rules_mg_large_adf54.dat -l -v

followed by running adas808 on all of these, firstly in both ic and ca mode for the “normal” size:

> run_adas808.sh -l mg 12 magnesium 1 12
/home/<user>/work/<normdir>
/home/<user>/adas/adf54/promotion_rules_mg_adf54.dat

and then in ca only for the “large”:

> run_adas808.sh -l mg 12 magnesium 1 12
/home/<user>/work/<lrgedir>
/home/<user>/adas/adf54/promotion_rules_mg_large_adf54.dat

Once this has been done, it is desireable to create an ADF11 file for the whole element. To do this requires recalculating the individual $\mathcal{PLT}$s for each stage on the same T_e, N_e grid. The IDL procedure **adas8xx_opt_prep_make_adf11.pro** will adjust the ADF42 driver files which specify the temperature, density and wavelength ranges for the PLT. A default temperature and density grid is provided, but a custom one can be used by specifying the *plasma* parameter, which is a structure identical to that shown in section 2.2.

```
IDL >  normdir='/home/<user>/work/<normdir>'
IDL >  largedir='/home/<user>/work/<lrgedir>'
IDL >  z_nuc=12
IDL >  adas8xx_opt_prep_make_adf11,        normdir=normdir, $
                                           largedir=largedir, $
                                           z_nuc=z_nuc
                                           plasma=plasma (optional)
```

Once this is finished, a Perl script will run adas810, to calculate the $\mathcal{PLT}$ for the 3 different couplings:

```
> run_optimise_plt.pl /home/<user>/work/<normdir> /home/<user>/work/<lrgedir>
  12 1
```

where the inputs are:

1 Directory where “normal” size runs were conducted
2 Directory where “large” size runs were conducted
3 Atomic number
4 Switch to use loadleveller (1=yes, 0=no)

Finally, once this is complete, the IDL routine **adas8xx_opt_make_adf11.pro** will combine the ADF11 files to produce the finished $\mathcal{PLT}$ for the element.

```
IDL >  normdir='/home/<user>/work/<normdir>'
IDL >  largedir='/home/<user>/work/<lrgedir>'
IDL >  z_nuc=12
IDL >  adas8xx_opt_make_adf11,             normdir=normdir, $
                                           largedir=largedir, $
                                           z_nuc=z_nuc
                                           a11file=a11file (optional)
```

By default the $\mathcal{PLT}$ file will be placed in /home/<user>/work/<normdir>/adf11/plt40/plt40_<elsymb>, but a new name/location can be specified by using the "a11file" keyword.

The ability to estimate approximately the wavelength of this missing emission in the more precise, but incomplete *ic* calculations is helpful. An example is shown in figure 2.11, which compares the feature photon emissivity coefficient for *ca* and *ic* resolution in the *normal* calculation with that for the *large ca* calculation. The effect of the extra configurations present in the latter can clearly be seen. While the *ca* approximation cannot be used for line spectroscopy, it can be used to gauge the approximate spectral regions in which spectral features are likely to appear and potentially confuse interpretation.

The *adf54* data sets following, archived for information in central ADAS

/home/adas/adas/adf54/promotion_rules_w.dat
/home/adas/adas/adf54/promotion_rules_w_large.dat

were prepared with **run_opt_808_offline.sh** for the *normal* and *large* studies of tungsten respectively.

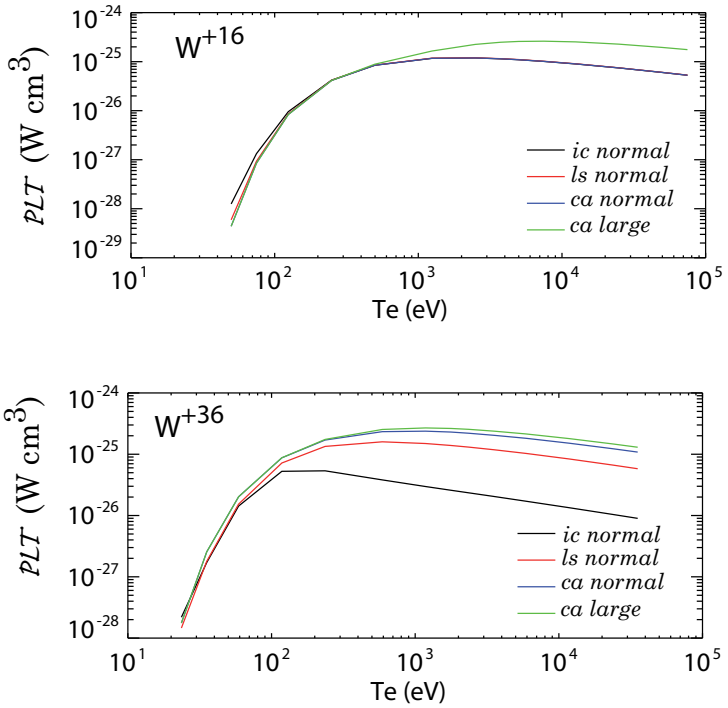

Figure 2.10: The total radiated power calculations for different resolutions. For most stages there is good agreement between the different resolutions as shown in the upper figure. However for those stages with open *d* and *f* shells, the *ca* and to a lesser extent the *ls* resolutions overestimate. The difference between the *large* and *normal ca* calculations remains a valid correction to the *normal ic* power.

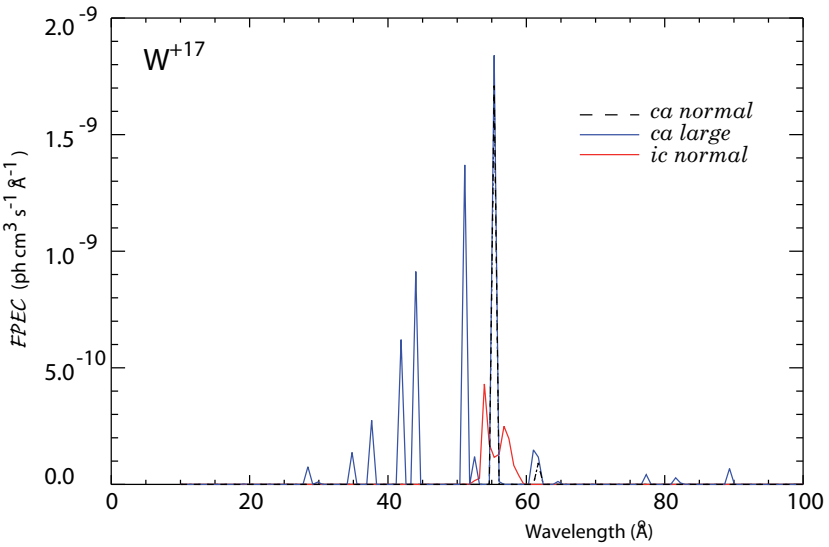

Figure 2.11: The feature emissivity coefficients obtained at the different resolutions and calculation sizes for W^{+17} at electron temperature T_e = 419 eV and electron density $N_e = 10^{13}$ cm^{-3}.

Chapter 3

Ionisation state of heavy elements

Effective ionisation and recombination coefficients linking the ionisation stages of heavy species are required. Obtaining these coefficients with precision, including their dependence on electron density as well as electron temperature, is a major task. As elaborated in *GCR - paper I*, the task is correctly carried out within the population structure calculations for each ionisation stage and requires the implementation of a representation of the infinite level atom. The procedures of condensation and projection, coupled with the independent generation of high quality state selective individual coefficients to very highly excited levels, are beyond the scope of the heavy species baseline. Following the route of simplification of the full collisional-radiative complexity as outlined in section 1.2, a problem remains in that the restricted number of key ionising and recombining reactions (such as the direct ionisation reaction from the ground state of a complex ion) are difficult to calculate with precision. Again a path must be followed trading universal coverage against precision. The ADAS heavy species provision in the ionisation and recombination case has three precision layers, namely a basic level exploiting general parametric forms, an intermediate level of individualised calculations of moderate precision (with a capacity for global coverage in the medium term) and a high level which will only be available for key selected ions (as identified in the procedures of chapter 4). We seek in the first instance to build the effective ionisation and recombination coefficients for the basic level entirely from atomic data available in the automatically generated *adf04* files of chapter 2. Then additional ADAS data formats *adf23*, *adf32* & *adf56* (for electron impact ionisation) and *adf09*, *adf46* & *adf55* (for dielectronic recombination) are introduced which can take the modelling to the intermediate level. These are closely related to *adf04*, *adf34* & *adf54* (for structure and emission) and the computational procedures for their creation will be seen to have strong resemblance and links to those of chapter 2. Current manpower resources suggest that it will be sometime before intermediate level coverage is global, but even following short of that goal, the intermediate results do enable optimised global scaling improvement of the basic level results. The globally-scaled basic level data is the ADAS *baseline* for ionisation and recombination. Discussion of the ADAS ambitions for reaching the highest level is reserved for chapter 5.

The construction of ionisation, recombination and power coefficients has been done extensively in the past, by exploiting simple parametric forms for key rates, such as the Lotz (1965, 1972,1973) formula for ionisation rates (based on the scaled Thomson (1912) classical ionisation cross-section expression), the general formula for dielectronic recombination of Burgess (1965,1965) and the excitation formula of Van Regemorter (1962). Prior to the use of more sophisticated, numerical tabulated, collisional- (and generalised-collisional-) radiative coefficients (for light elements), impurity transport codes such as STRAHL(Behringer, 1984), SANCO (Taroni *et al* , 1995) relied exclusively on these fomulae, which in turn required only relatively simple parameters such as oscillator strengths, ionisation potentials and excitation energies. For quick estimates on unfamiliar species, such methods are still in use. Within the ADAS Project, the old parameterisations are made available (see ADAS data format *adf03* and the ADAS user manual) and are called 'Case A' parameterisations. These parametric forms are of only modest precision in general, depending partly on the quality of the parameters themselves, but are unsafe for medium/heavy species. They are the starting point for the next subsections.

3.1 Ionisation

3.1.1 Parametric forms

The principal ADAS semi-empirical expression for the electron impact ionisation rate coefficient is that of Burgess and Chidichimo (1983) $S^{bchid}_{shd}(z,\chi,\zeta;T_e)$. viewed as a formula for the direct ionisation of an ion of charge z from a quantum shell (*shell direct* ionisation, with generic name S_{shd}) of ionisation potential χ and number of equivalent electrons in the shell, ζ. The expression is

$$S^{bchid}_{shd} = 2\sqrt{\pi}\alpha c a_0^2\, C\, \zeta (I_H/\chi)^{3/2}(\chi/kT_e)^{1/2} \times E_1(\chi/kT_e) w \tag{3.1}$$

where $w = \{\ln(1 + kT_e/\chi)\}^{\beta(1+kT_e/\chi)}$ and $\beta = [\{(100z + 91)/(4z + 3)\}^{1/2} - 5]/4$. The constant $2\sqrt{\pi}\alpha c a_0^2 = 2.17 \times 10^{-8}$ cm^3 s^{-1} and the multiplier $C = 2.3$ (as recommended by Burgess and Chidichimo). Although the expression has similarities to the Lotz formula, which is called S^{lotz}_{shd} here, the latter was created before a proper recognition of the contribution of excitation/autoionisation to net ionisation. Nonetheless ADAS has a legacy arising from the need historically to reproduce the Lotz results. Some further comments on Lotz are appropriate here. The basic Lotz expression is for shell-direct ionisation. However the default parametric form (a single multiplying parameter) was modified for the special cases of ionisation of ions of charge state 0-3 with nuclear charges $z_0 \leq 30$. Also in application of the formula, Lotz specified the shells to be included - restricting to at most three. ADAS called the use of Lotz, Case A, and the ADAS data format *adf03* (to be discussed later) has data for evaluation of ionisation in Case A.

Returning to the preferred S^{bchid}_{shd}, the IDL procedure is **r8fbch.pro**. It requires the ionisation potential in Rydbery energy units (I_H) and the temperature in Kelvin, delivering the ionisation rate coefficient in units of $cm^3 s^{-1}$. In fact, ADAS also makes use of another expression, $S^{ecip}(z,\chi,\zeta;T_e)$, called the *Exchange-Classical-Impact-Parameter* approximation (Burgess and Summers, 1972) with IDL procedure **r8necip.pro**. For example, for ionisation from the valence shell $2p^2$ of O^{+2}, type at the IDL command line

```
IDL>te=adas_vector(low=1.0e5,high=1.0e7,num=10)
IDL>iz = 2
IDL>xi= 4.038
IDL>zeta=2
IDL>result=r8fbch( iz = iz, xi = xi, zeta = zeta, te = te)
IDL>plot,te,result,/xlog,/ylog
IDL>result=r8necip( iz = iz, xi = xi, zeta = zeta, te = te)
IDL>oplot,te,result
```

S^{ecip} is mostly reserved for ionisation out of the highly excited states of ions which matter in full collisional-radiative modelling, and need not concern us further here. It is S^{bchid} which has been examined and tuned for ionisation of atoms and ions in their ground states. Since $S^{bchid}_{shd}(z,\chi,\zeta;T_e)$ gives the direct ionisation rate coefficient out of one *nl* shell, the total direct ionisation rate out of all the shells of a complex atom or ion in its ground configuration (state), which is called the *configuration shell direct* ionisation, is

$$S^{bchid}_{cfg-shd} = \sum_{i=1}^{N_s} S^{bchid}_{shd}(z,\chi_i,\zeta_i,T_e) \tag{3.2}$$

Here the ground configuration is $n_1 l_1^{q_1} \cdots n_i l_i^{q_i} \cdots n_{N_s} l_{N_s}^{q_{N_s}}$, $\zeta_i \equiv q_i$ and χ_i is the binding energy of an electron in the shell i. It is to be noted that such an expression is really only intended to apply to ionisation from the ground state or possibly metastable states of an ion. Ionisation out of highly excited states is completely dominated by valence electron loss alone to an adequate approximation. Also it has not been specified if the ionisation energy of the valence shell electron is the ionisation potential of the ground state or the configuration average valence electron orbital binding energy. Since the threshold region in temperature of the ionisation rate coefficient is important for ionisation balance, in fact the ionisation potential is to be preferred for the valence electron shell, but the above formulation still has a major omission, namely the contribution of *excitation-autoionisation*. The latter must be included before the total ionisation rate coefficient is realistic for heavy species. The excitation-autoionisation contribution from the ground

state of an ion (ignoring multiple electron shake-off and shake-down) for promotion from a particular shell (generic name S_{ea}) can be written as

$$S_{ea} = \sum_{k=1}^{N_r} q_k(T_e) \frac{A_k^a}{A_k^a + A_k^r} = \sum_{k=1}^{N_r} q_k(T_e) B_k \tag{3.3}$$

where the sum is over resonance states k, with q_k the excitation rate coefficient to resonance k from the ground state and B_k is the branching ratio for autoionisation, assembled from A_k^a the autoionisation probability and A_k^r the radiative stabilisation probablility. For most ions, the details of equation 3.3 for all the contributing resonances are not available or feasible to evaluate. It was a prescription for the inclusion of excitation/autoionisation in expressions of the same form as equation 3.2 by Burgess *et al* (1977) (elaborated by Burgess and Chidichimo (1983)), which allowed simple semi-empirical expressions to match higher quality results with significantly lower standard error. This is especially important for heavy species ions for which the S_{ea} contribution often dominates shell direct ionisation. Burgess

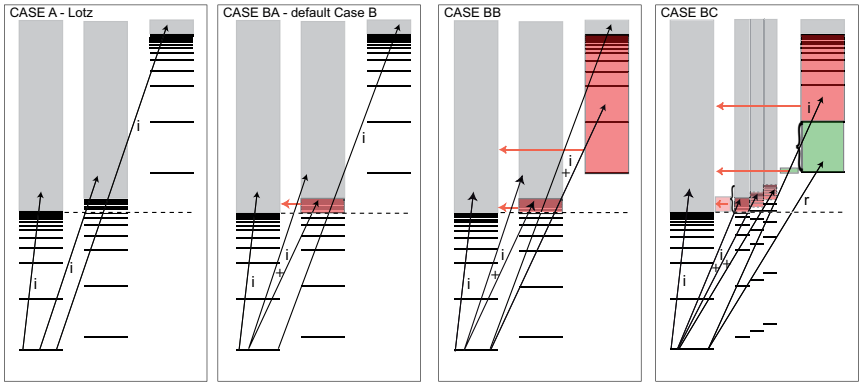

Figure 3.1: Approaches to semi-empirical formula for ionisation from the ground configuration of ion. CASE A includes only shell direct ionisation and is evaluated in the Lotz approximation; CASE BA includes Burgess-Chidichimo type ii autoionisation in shell direct ionisation by ζ and χ adjustment; CASE BB includes Burgess-Chidichimo type i autoionisation in shell direct ionisation by ζ and χ adjustment; CASE BC separates Burgess-Chidichimo type i into a lowest explicit resonance and the rest in shell direct ionisation by ζ and χ adjustment. The explicit resonance contribution (an excitation rate coefft. with finite threshold) is normalised to its equivalent type i ionisation share. Additionally, the contribution of the type ii autoionisation is diluted for partially filled d and f shells of heavy species ion since there are many series of rising thresholds - weakening the type ii assumption. CASE BC also includes the option of global scaling of ionisation and resonance groups. The symbol 'i' or 'r', beside the black arrows, indicates whether the contribution is handled as an 'ionis.' group or as a 'reson.' group - see later text. The continua available to the directly ionised electron are shown in grey. Auto-ionising resonances replaced by equivalent continuum are shaded in red. Green shows the lowest type (i) resonance and the equivalent continuum used to evaluate it.

et al (1977) distinguished two situations: (i) where the first autoioning resonance of a series lies well above the first ionisation threshold - expected with excitations from deeper principal quantum shells than that of the valence shell; (ii)where the autoionising series of resonances lie densely through the first ionisation threshold - expected with excitations from the same principal quantum shell as the valence (orbital) shell; On the assumption of unit branching ratio for all resonances and adopting a Correspondence Principle view, summations over resonances were replaced by integrals over energy, leading simply to shell-direct-like ionisation coefficients but with modified threshold energies χ' and numbers of equivalent electrons ζ'. Parametric forms in ADAS for heavy species extend the Burgess approach as summarised in figure 3.1. Each sub-panel shows the ordinary and the type (ii) and (i) auto-ionising level structures. The auto-ionising levels lie above the dashed first threshold line. ADAS Case BA, for historical reasons and to be able to generate the $S^{bchid}_{cfg-shd}$ equivalent of $S^{lotz}_{cfg-shd}$ neglected type (i). Case BB includes type (i) but only via an ionisation threshold reduction. Both Cases BA and BB are evaluated as a simple sum of effective shell direct ionisations driven by a list of shell occupancies and ionisation potentials (real or effective). The case distinctions are sinply ADAS working practice. The new Case BC introduced here dilutes the number of equivalent electrons included in type (ii) for d and f shells and includes an explicit excitation rate coefficient for the lowest resonance of each type (i) series. The latter, evaluated in the spirit of the Burgess-Chidichimo assembling from shells, gives a contribution to the ionisation which

may be written as

$$S^{bchid}_{cfg-ea} = \sum_{k=1}^{Nser_{ii}} S^{bchid}_{ea}(z, \chi^{exc}_k, \zeta_k, T_e) \tag{3.4}$$

Note that this is not the whole excitation-auotionisation, most of which is subsumed in the modified shell direct part. Finally the total ionisation rate coefficient from a configuration becomes

$$S^{bchid}_{cfg-tot} = \sum_{i=1}^{N_s} S^{bchid}_{shd}(z, \chi'_i, \zeta'_i, T_e) + \sum_{k=1}^{Nser_{ii}} S^{bchid}_{ea}(z, \chi^{exc}_k, \zeta_k, T_e) \tag{3.5}$$

To evaluate these parametric forms for an ion, the ground state configuration specification is required along with the orbital binding energies for each of the shells of the configuration and the true ionisation potential. ADAS can provide these data for any ion of any element. As has been discussed in section 1.1, configurations and ionisation potentials of ground states of ions are in *adf00* data sets accessible from IDL with **read_adf00.pro**. The shell ionisation energies are obtainable with an additional procedure **config_orbital_energies.pro**. To complete the utilities in ADAS for ionisation studies, attention is drawn to the procedure **tev_alf_s.pro**. This gives an approximate central electron temperature in eV at which the ionisation rate coefficient for an ion is required in ionisation balance. More precisely it gives the temperature at which the ionisation coefficient $S(z \rightarrow z+1)$ is equal to the recombination coefficient $\alpha(z+1 \rightarrow z)$. Precision in ionisation rate coefficients is important principally within one tenth and ten times this value. In illustration, at the IDL command line type

```
IDL>z0=73)
IDL>z_ion = 6
IDL>read_adf00,z0=z0,z_ion=z_ion,config=config,ionpot=ionpot
IDL>print,'config=',config
IDL>config_orbital_energies,z0,z_ion,fulldata=fd
IDL>print,'fd.config=',fd.config
IDL>print,'fd.config_n=',fd.config_n
IDL>print,'fd.config_l=',fd.config_l
IDL>print,'fd.config_energy=',fd.config_energy
IDL>tev=tev_alf_s,z_ion,ionpot
IDL>print,'tev=',tev
```

Note that **config_orbital_energies.pro** has an optional *config* keyword parameter for input of a configuration of the user's choice. It overrides acquisition of the ground configuration from the appropriate *adf00* data set.

Expression 3.5 may be applied to any heavy element ion and is suitable for the ADAS *baseline* data production. It gives a significant improvement for heavy species ions over earlier approximations. In subsection 3.1.2 we consider more sophisticated (but currently less complete) ab initio calculation of ground state ionisation coefficients. Before doing so, it is clear that comparison of $S^{bchid}_{cfg-tot}$ with the results of refined calculation might suggest improvements. In particular we might expect some further adjustment of the multiplying constant C in equation 3.1 as a function of z_0, z and shell. It is appropriate and safest to limit such adjustments at this stage. In ADAS we introduce grouping of shells in shell direct ionisation and of type i autoionising resonances which we call *ionis.* groups and *reson.* groups with an adjustable multipler for each group. The resulting approximate form, as given in equation 3.6 below, is suitable for optimised fitting to improved data.

$$\begin{aligned} S^{approx}_{cfg-tot} = & \sum_I c_I \sum_{i \in I} S^{bchid}_{shd}(z, \chi'_i, \zeta'_i, T_e) \\ & + \sum_R c_R \sum_{k \in R} S^{bchid}_{ea}(z, \chi^{exc}_k, \zeta_k, T_e) \end{aligned} \tag{3.6}$$

where the summations are over the set of quantum shell *ionis.* groups I and quantum shell *reson.* groups R and the members of each group, where the extra scaling parameters c_I and c_R are expected to be of of order unity. In practice in central ADAS, we limit to two *ionis* groups, namely the valence shell and all other shells, and to one *reson.* group.

The principles discussed above are perhaps clearer from two examples. Consider an ion, such as W^{+12}, whose ground state has the outer electron configuration $5s^25p$. The ionisation potential for the $5p$ electron is I_{5p} and for a $5s$ electron is I_{5s}. The initial *shell direct* equivalent electron assignments are $\zeta_{5p} = 1$ and $\zeta_{5s} = 2$ at these ionisation potentials respectively. However autoionising levels of the form $5s5pnl$ lie densely through the $5s^2$ ionisation threshold and on into the $5s^2 + e$ continuum. Excitation of a $5s$ electron to such levels leads to auto-ionisation into this continuum with almost unit branching probability. The effect can be included in the *shell direct* part by setting the effective ionisation potential for the $5s$ electron to I_{5p}. This is the case ii situation. Case A parameterisation includes only the outer two shells, however the illustrative ion has the complete shell structure $1s^22s^22p^63s^23p^63d^{10}4s^24p^64d^{10}4f^{13}5s^25p$. The *shell direct* part from the inner shells, especially $4f^{13}$ and $4d^{10}$ have a large ζ weighting and must be included. The first auto-ionising configuration from promotion of a $4f$ electron is $4f^{12}5s^25p^2$ and it is noted that it lies substantially above the $5s^2$ ionisation threshold. It might be appropriate to include such auto-ionisation by reducing the ionisation potential of the $4f$ electron from I_{4f} to $I_{4f} - I_{5p}$. This is the case i situation and Case B implementation. On the other hand, ionisation cross-sections are zero at theshold whereas excitation cross-sections (for ions) are finite at threshold. Detailed experimentally measured ionisation cross-sections show steps at discrete auto-ionising level energies. For a more precise description within the Case C parameterisation, we include an S^{bchid}_{cfg-ea} contribution from $4f^{13}5s^25p \rightarrow 4f^{12}5s^25p^2$ and then put autoionising configurations $4f^{12}5s^25pnl$ with $n > 5$ into the *shell direct* part at effective ionisation potential $I_{4f} - I_{6s}$.

An IDL procedure **sbchid_cfg_tot.pro** evaluates equation 3.5. An example is shown below for W^{+12}

```
IDL>z0_nuc=74)
IDL>z_ion = 12
IDL>tev=tev_alf_s,z_ion,ionpot
IDL>low=11605.4*tev/100
IDL>high=11605.4*tev*100
IDL>te=adas_vector(low=low,high=high,num=21)
IDL>sbchid_cfg_tot,z0_nuc,z_ion,te=te,coef=coef
IDL>print,'te=',te
IDL>print,'coef=',coef
```

sbchid_cfg_tot.pro has a number of other non-positional parameters. These include specifying the initial configuration, a non-default *case* and so on. Such a more directed illustration is given below. The full specification is in Appendix B.

```
IDL>z0_nuc=74)
IDL>z_ion = 12
IDL>config_orbital_energies,z0,z_ion,fulldata=fd
IDL>config_z=fd.config[z_ion]
IDL>config_z1=fd.config[z_ion+1]
IDL>ionpot_z=fd.ionpot[z_ion]
IDL>excpot_z1=0.0
IDL>case_ionis='case bc'
IDL>ci_v= 1.1
IDL>cr= 0.9
IDL>tev=tev_alf_s,z_ion,ionpot
IDL>low=11605.4*tev/100
IDL>high=11605.4*tev*100
IDL>te=adas_vector(low=low,high=high,num=21)
IDL>sbchid_cfg_tot,z0_nuc,z_ion,te=te,coef=coef,
    ionpot_z=ionpot_z, excpot_z1=excpot_z1, case=case,ci_v=ci_v,cr=cr
IDL>print,'te=',te
IDL>print,'coef=',coef
```

3.1.2 Configuration average distorted wave ionisation

The ADAS Project, in collaboration with Auburn University (Dr. Stuart Loch), has had in development for some time more sophisticated calculations of electron ionisation rate coefficients. The most powerful methods (see chapter 5) are still limited to few electron systems and so the distorted wave method is the main *ab initio* method which can assist in a first 'lift' of the heavy species baseline. The development of the method in the configuration average *ca* approach by Pindzola, Griffin and Bottcher is particularly valuable. It has reasonable economy of computation, while allowing acccess to complex, multi-electron ions, highly excited states, excitation autoionisation and radiative damping. That is to say it is able to evaluate the four key constituent reactions, namely, *shell direct ionisation*

$$(n_1 l_1)^{q_1} k_i l_i \rightarrow (n_1 l_1)^{q_1} k_e l_e k_f l_f,$$

with differential(in ejected electron energy) cross-section

$$\frac{d\sigma_{ion}}{d\epsilon} = \frac{32}{k_i^3 k_e k_f}(q_1 + 1) \sum_{l_i, l_e, l_f} (2l_i + 1)(2l_e + 1)(2l_f + 1) M(ef; 1i); \tag{3.7}$$

the *shell excitation*

$$(n_1 l_1)^{q_1+1} (n_2 l_2)^{q_2-1} k_i l_i \rightarrow (n_1 l_1)^{q_1} (n_2 l_2)^{q_2} k_f l_f,$$

with cross-section

$$\sigma_{excit} = \frac{8\pi}{k_i^3 k_f}(q_1 + 1)(4l_2 + 3 - q_2) \sum_{l_i, l_f} (2l_i + 1)(2l_f + 1) M(2f; 1i); \tag{3.8}$$

the *Auger breakup*

$$(n_1 l_1)^{q_1} (n_2 l_2)^{q_2} (n_3 l_3)^{q_3} \rightarrow (n_1 l_1)^{q_1+1} (n_2 l_2)^{q_2-1} (n_3 l_3)^{q_3-1} k_e l_e,$$

with autoionisation rate coefficient

$$A^a = q_2 q_3 \frac{(4l_1 + 2 - q_1)(4l_2 + 2) M(1e; 23)}{k_e}; \tag{3.9}$$

and finally the competing *radiative damping*

$$(n_1 l_1)^{q_1-1} (n_2 l_2)^{q_2} \rightarrow (n_1 l_1)^{q_1} (n_2 l_2)^{q_2-1},$$

with spontaneous emission coefficient

$$A^r = \frac{8\nu^3}{3c^2} \frac{q_2(4l_2 + 3 - q_2) D(12)}{(4l_1 + 2)(4l_2 + 2)}, \tag{3.10}$$

which are assembled to yield expressions for $S_{cfg-shd}$ and S_{cfg-ea} (see equations 3.2 and 3.3) with appropriate integration over ejected electron and final colliding electron energies and Maxwellian averaging over initial projectile elctron energy. In the above k_i, k_f and k_e denote initial and final projectile wave numbers and ejected electron wave number respectively. $M(14; 23)$ denotes the squared two-body Coulomb matrix element and $D(12)$ the one-body dipole matrix element. Other notation is conventional.

The broad procedural approaches of section 2.1 are applicable but the promotional rules for determining ionisation and excitation-autoionisation configurations and the computational procedures for the quantal structure (Cowan) and cross-section calculations are sufficiently different to necessitate a parallel system. The underlying configuration average distorted wave code is executed online by the code ADAS802 (paralleling ADAS801) and offline by ADAS8#2. Drivers belong to ADAS data format *adf32*. Although these drivers appear as concatenations of *adf34* drivers, the control parameters are differently organised and there is some additional information present. An *adf32* data set will not work with ADAS801. The drivers can be prepared using promotional rules. The promotional rules for ionisation/excitation-autoionisation for all possible ground states of ions of elements are in data format *adf56* (equivalent to *adf54* for structure). An *adf56* data set is read in IDL by **read_adf56.pro** and the specific rules for

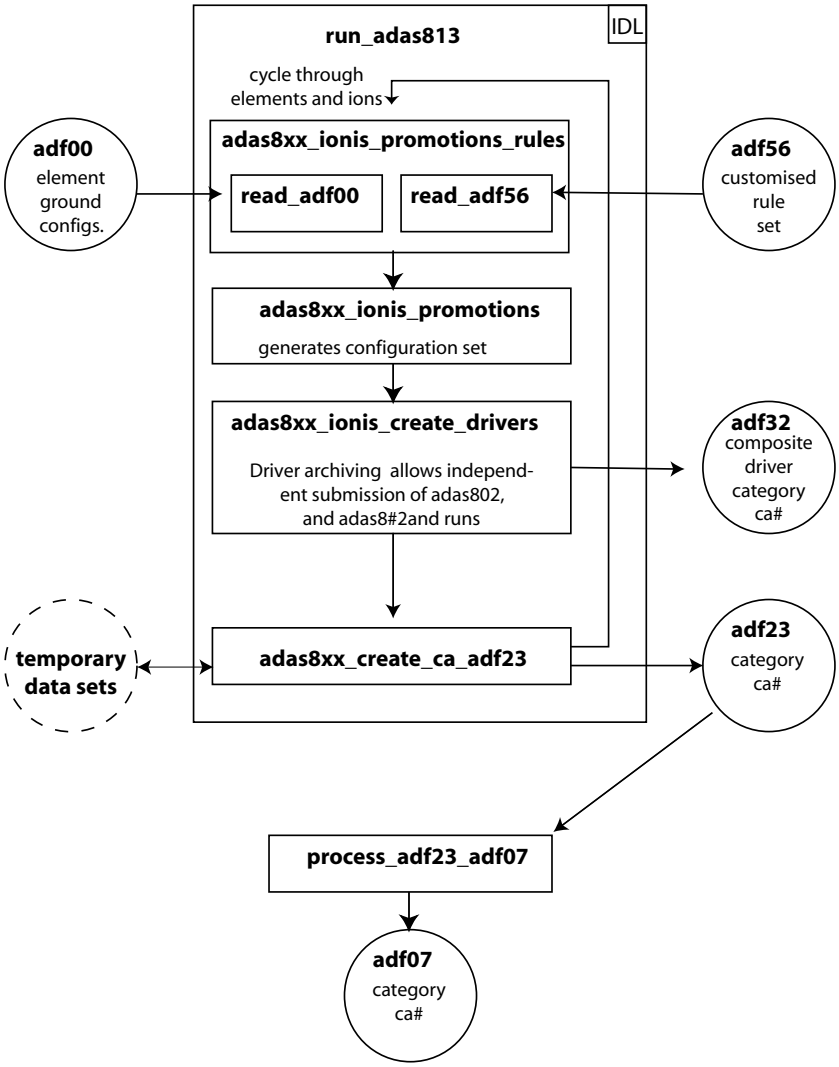

Figure 3.2: Schematic of **run_adas813.pro** program flow for complete sequential heavy species calculation from the IDL command line. The various temporary and permanent data sets created are indicated. Note that the **adas8xx_create_ca_adf23.pro** step spawns an **offline_adas** script. As shown, the simpler directly usable ADAS format *adf07* of *cfg-tot* ionisation rate coefficients may be created from an *adf23* data set.

an element extracted by **adas8xx_ionis_promotion_rules.pro**. The procedure **adas8xx_ionis_promotions.pro** establishes the actual configurations for both direct ionisation and excitation-autoionisation and the drivers are created by **adas8xx_ionis_write_drivers.pro**. Running **adas8xx_ionis_promotion_rules.pro** calls all three IDL procesdure to generate the *adf32* file. After this calling the PERL routine **adas8#2.pl** generates the final *adf23* data set.

In practice, large scale calculations are initiated with the **run_813_offline.sh** shell script which distributes the computation amongst many processors for offline execution. This is managed at JET by LOADLEVELLER.

index[]	:	index of ground configuration of each ion of element in *adf56* file
config[]	:	ground conf[]iguration for each ion of element
n_el[]	:	number of electrons for each ion of element
no_v_shl[]	:	number of shells to treat as valence shells. Max. 2 relevant to relating ion and parent.
v1_shl[]	:	first valence shell position in adf56 configuration specifications.
v2_shl[]	:	second valence shell position in adf56 configuration specifications. zero if none defined.
drct_eval_v[]	:	evaluate direct ionisation from the valence shell(s).
drct_eval_cl[]	:	evaluate direct ionisation from other non-valence (closed) shells.
min_shl_cl[]	:	lowest closed shell to include (position in adf56 configuration specifications).
exca_eval_v2[]	:	evaluate excitation/autoionisation from second valence shell if identified.
max_dn_v2[]	:	maximum change in v2 n-shell to be included.
min_dn_v2[]	:	minimum change in v2 n-shell to be include.
max_dl_v2[]	:	maximum change in v2 l-shell to be included.
min_dl_v2[]	:	minimum change in v2 l-shell to be include.
exca_eval_cl[]	:	evaluate excitation/autoionisation from other non-valence (closed) shells.
max_dn_cl[]	:	maximum change in closed n-shell to be included.
min_dn_cl[]	:	minimum change in closed n-shell to be included.
max_dl_cl[]	:	maximum change in closed l-shell to be included.
min_dl_cl[]	:	minimum change in closed l-shell to be included.
exst_eval[]	:	evaluate ionisation from excited states.
exst_adf00_prt[]	:	assume parent for building excited states is as present in the adf00 data set for the ion.
exst_prt_hole_shl[]	:	specify position of shell in ground configuration to form parent if not from adf00 above.
max_n_exst[]	:	maximum n-shell for excited states to be included.
max_l_exst[]	:	maximum l-shell for excited states to be included.
drct_eval_exst_v[]	:	evaluate direct ionisation from excited state valence shells.
drct_eval_exst_cl[]	:	evaluate direct ionisation from excited state non-valence (closed) shells.
exca_eval_exst_v[]	:	evaluate excitation/autoionisation for excited states from valence shells (v1 and v2 above).
exca_eval_exst_cl[]	:	evaluate excitation/autoionisation for excited states from non-valence (closed) shells.

To make full use of these capabilities, we note the following and draw attention to the close similarity to section 2.3. The reference promotion rule set, as archived in an *adf56* data set, is again handled as an IDL structure *ref_rules* (with vectors of length 180 spanning all possible ground states). In the ionisation case, it is defined as shown above along with the meaning of each rule. The rules are somewhat different from those of *adf54*, especially the role of valence shells. In the ionisation context, two valence shells are used to deal with ground configurations (and assigning parents) of neutral and near-neutral ions where an active single active shell may be ambiguous. For inner shell direct ionisation and excitation/autoionisation of excited states, the rules apply as for the shells of the ground state. This maintains proper consistency with ground state ionisation.

The *plasma* structure contains only electron temperature data in the ionisation rate coefficient case since there is no density or spectrometer data required. It is defined below

theta[]	:	electron temperature vector(K)
indx_theta[]	:	index vector sub-selection from full *theta* vector
unscaled_theta	:	1 (or set)=> temperatures *theta* are not scaled with z_1 0 (or not set) => temperatures *theta* are scaled with z_1

The ADAS standard for data set naming in the ionisation case is as shown in table 3.1.2. The temporary datasets and principal loops in execution of the offline **run_813_offline.sh** script are entirely equivalent to those of **run_808_offline.sh** shown in figure 2.5. The organisation and disposition of the data sets are of course different and are summarised in figure 3.3. In the following we illustrate the various capabilities at the IDL command line. Firstly we read an *adf56*

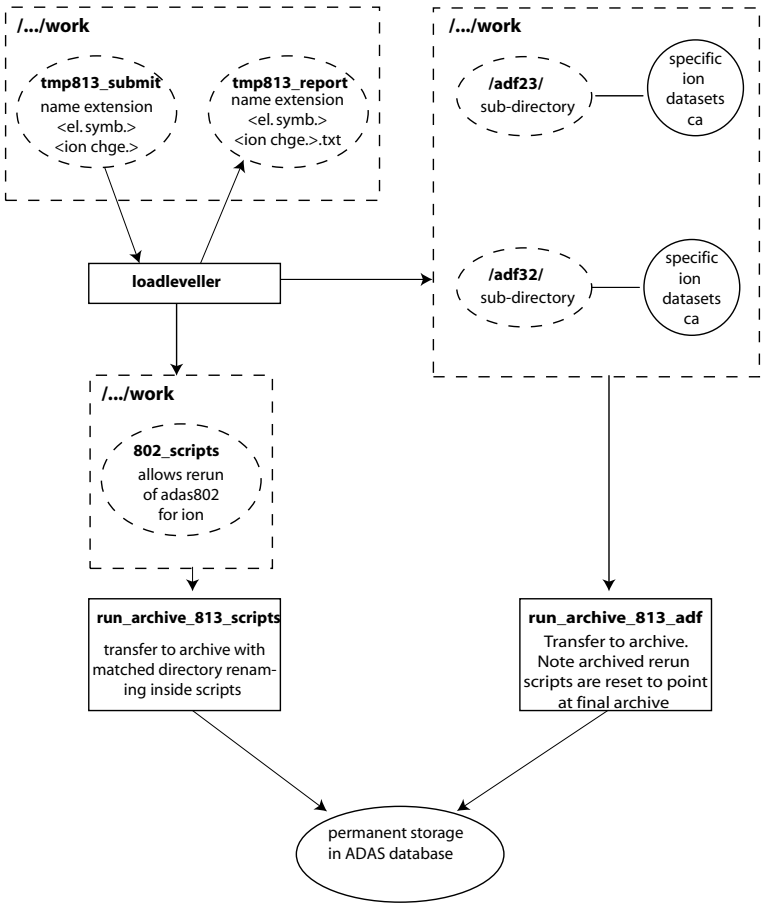

Figure 3.3: Organisation and disposition of data sets used and created in distributed off-line execution managed by LOADLEVELLER

adf no.	subdir.	subsubdir.	datasets
adf07/	copmm#<nuc.chge.> /		ca#<el.symb.><ion chge.>.dat
			ca#<el.symb.><ion chge.>_t1.dat
			ls#<el.symb.><ion chge.>.dat
			ic#<el.symb.><ion chge.>.dat
adf23/	<class>06		<class>06_<el.symb.>_<prt.layer>.dat
			ls#<el.symb.><ion chge.>.dat
			ls#<el.symb.><ion chge.>.dat
adf32/	<elem.name> /		<el.symb.><ion chge.>.dat
			<el.symb.><ion chge.>_inst.dat
			<el.symb.><ion chge.>_ls_pp.dat
			<el.symb.><ion chge.>_ic_pp.dat
scripts/	<elem.name> /		<el.symb.><ion chge.>_ls_801_script
			<el.symb.><ion chge.>_ic_801_script
			<el.symb.><ion chge.>_ca_810_script
			<el.symb.><ion chge.>_ls_810_script
			<el.symb.><ion chge.>_ic_810_script

Table 3.1: text required

data set of the ionisation promotion rules set :

IDL> *a56file = '/home/adas/adas/adf56/promotion_rules_large.dat'*
IDL> *read_adf56,file=a56file,ref_rules=ref_rules*
IDL> *print,ref_rules.config[0]*
IDL> *print,ref_rules.min_l_cl[0]*

The actual ADF32 driver file for a specific case are obtained as:

IDL> *z_nuc=36*
IDL> *a56file = '/home/adas/adas/adf56/promotion_ionis_rules_large.dat'*
IDL> *z_ion=13*
IDL> *wkdir=<working_directory>*
IDL> *adas8xx_ionis_promotion_rules, z_nuc = z_nuc,*
a56file = a56file,
z_ion = z_ion,
wkdir = wkdir

Finally the *adf23* data set is created using the *adf32* driver, through the Perl script, adas8#2.pl

> *adas8#2.pl <adf32file> <adf23file>*

3.2 Recombination

It can be assumed that radiative recombination and dielectronic recombination are independent processes for heavy species modelling with negligible error in comparison with other atomic structure and rate coefficient uncertainties. Also because of the strong z-scaling of effective electron density for collisional-radiative effects and sensitivity to electron temperature ($\sim T_e^{-9/2}$), three-body recombination (the inverse of electron impact ionisation) is also negligible, except possibly for neutral species at very low temperatures. So the effective recombination coefficient may be written as

$$\alpha^{(z+1\rightarrow z)} = \alpha^r_{eff} + \alpha^d_{eff} \tag{3.11}$$

Instead of detailed modelling of redistribution and ionisation from excited states, which reduces particularly the effective dielectronic rate coefficient in finite density plasma (radiative recombination is predominantly to low levels and so is less affected), a cut-off n-shell, n_t, may be introduced (Wilson, 1960).

$$n_t = [5.57 \times 10^{17}(\mathrm{cm}^{-3}/N_e)z_1^6(kT_e/I_H)^{1/2}]^{1/7} \tag{3.12}$$

Captures to levels below n_t are assumed to populate ultimately the ground level and so contribute to the effective coefficient, whereas captures to levels above n_t do not. n_t depends on electron density and it is this which causes the finite density collisional-radiative effects. This is acceptable for heavy species ions at normal tokamak densities and in fact, again because of the z-scaling of effective electron density, the effective recombination coefficient is quite closely converged on its low density limit under such conditions. We return to this point in section 3.3

3.2.1 Radiative recombination

The effective radiative recombination coefficient is a sum over capture to ground and excited state quantum shells up to the cut-off principal quantum shell n_t. That is

$$\alpha^r_{eff} = \sum_{l \geq l_g} \alpha^r_{n_g l} + \sum_{n,n>n_g;l,l<n}^{n_t} \alpha^r_{nl} \tag{3.13}$$

where the basic shell-selective free-electron capture rate coefficient is

$$\alpha^r_{nl}(T_e) = \frac{8\alpha^4 c}{3\sqrt{3}\pi a_0}\left[\frac{8\pi a_0^2 I_H}{kT_e}\right]^{3/2} \frac{(2l+1)f_{ph}\langle g^{II}_{nl}\rangle}{n^2\nu^3_{nl}} e^{I_{nl}/kT_e} E_1(I_{nl}/kT_e) \tag{3.14}$$

ν_{nl} ia the effective principal quantum number for the nl-shell, $I_{nl} = z_1^2 I_H/\nu^2_{nl}$ is the ionisation potential for the shell and f_{ph} is a 'phase factor' taking account of the shell occupancy (or fractional parentage factors). We introduce the Maxwell exponential averaged bound-free Gaunt factor defined as

$$\langle g^{II}_{nl}\rangle = \frac{1}{E_1(I_{nl}/kT_e)} \int_{I_{nl}/kT_e}^{\infty} \frac{g^{II}_{nl} e^{-x}}{x} dx, \tag{3.15}$$

where g^{II}_{nl} is the usual bound-free Gaunt factor. We adopt a pure configuration, one-electron transition approximation - as for ionisation. Parametric forms result from the choice of f_{ph} and ν_{nl} and the treatment of $\langle g^{II}_{nl}\rangle$. Commonly $\langle g^{II}_{nl}\rangle = 1$ ('no Gaunt factor'), $\langle g^{II}_{nl}\rangle = \langle g^{II}_{nl}\rangle_H$ ('hydrogenic Gaunt factors') and $\langle g^{II}_{nl}\rangle = \langle g^{II}_{nl}\rangle_{DW}$ ('distorted wave Gaunt factors' - one-electron transitions in an adjustable model potential) are made. Also, with the assumption that ν_{nl} is independent of l, the 'bundle-n' approximation may be adopted. That is

$$\langle g^{II}_{n}\rangle = \sum_{l} \frac{(2l+1)\langle g^{II}_{nl}\rangle}{n^2} \tag{3.16}$$

Parametric forms

ADAS evaluates the above approximations using the methods of Burgess and Summers (1987). Two procedures are available for use at the IDL command line, namely **alf_r_bdn.pro** and **alf_r_bdnl.pro** for 'bundle-n' and 'bundle-nl' approximation respectively. Keywords include *approx* with character values *no-gf*, *h-gf* and *dw-gf* (*dw-gf* is not allowed for **alf_r_bdn.pro**). **alf_r_bdnl.pro** in 'dw-gf' mode requires a screening configuration, keyword: *config*, as a string in Cowan, standard or Eissner form. An electron temperature vector, *Te*, is required as an input parameter. An illustration is given below. The full specification of the procedures is in appendix B.

```
IDL> z0_nuc=36
IDL>z_ion=10
IDL>z1_ion=z_ion+1
IDL>n=20
IDL>tev=tev_alf_s,z_ion,ionpot
IDL>low=11605.4*tev/100
IDL>high=11605.4*tev*100
IDL>te=adas_vector(low=low,high=high,num=21)
IDL>alf_r_bdn,z0_nuc=z0_nuc,z_ion,n=n,te=te,coef=coef
IDL>print,'n=',n
IDL>print,'te=',te
IDL>print,'coef=',coef
IDL>
IDL>n=4
IDL>l=0
IDL>config_orbital_energies,z0_nuc,z_ion,fulldata=fd
IDL>config=fd.config[z1_ion]
IDL>ionpot=fd.ionpot[z_ion]
IDL>approx='dw-gf
IDL>alf_r_bdnl,z0_nuc=z0_nuc,z_ion,n=n,l=1,te=te,coef=coef,
config_z1=config_z1,ionpot_n=ionpot_n,approx=approx
IDL>print,'n=',n
IDL>print,'l=',l
IDL>print,'te=',te
IDL>print,'coef=',coef
```

The complete effective radiative recombination coefficient, α^r_{eff} may be assembled from these partial coefficients. The procedure is **alf_r_tot.pro**, which takes in an electron temperature vector, *te*, and an electron density vector, *dens*, and returns the coefficient array, *coef*. ADAS has historically provided case A and case B approximations for α^r_{eff}. These are set by the keyword *case_rr* which takes values *basic*, *case_a*, *case_b* and *case_c*. It is to be noted that all the ground configuration and shell ionisation potentials for heavy species are available from **config_orbital_energies.pro** and so the parameters of **alf_r_tot.pro** can be quite simple. They include keywords *z0_nuc*, *z_ion* and *case_rr*. If the electron density vector is omitted, the complete zero-density n-shell sum is returned. The definitions of the cases are as follow:

case A: 'bundle-n' for $n \geq n_g$; ν_{n_g} from *adf00* ion. pot.; $f_{ph} = 1$; *no-gf*
case B: 'bundle-n' for $n \geq n_g$; ν_{n_g} from *adf00* ion. pot.; $f_{ph} = 1 - q_{n_g}/n_g^2$; *h-gf*;
case C: 'bundle-nl' for $n = n_g$; $\nu_{n_g l}$ from *adf00* orb. ergys. & ion. pot.; $f_{ph} = 1 - q_{n_g l}/(2l+1)$; *dw-gf* for $n = n_g$; 'bundle-n' for $n > n_g$; *h-gf* for $n > n_g$
case basic: currently ≡ case B

```
IDL> z0_nuc=36
IDL>z_ion=10
IDL>z1_ion=z_ion+1
IDL>config_orbital_energies,z0_nuc,z_ion,fulldata=fd
IDL>config=fd.config[z1_ion]
IDL>ionpot=fd.ionpot[z_ion]
IDL>tev=tev_alf_s,z_ion,ionpot
IDL>low=11605.4*tev/100
IDL>high=11605.4*tev*100
IDL>te=adas_vector(low=low,high=high,num=21)
IDL>alf_r_tot,z0_nuc,z_ion,case_rr=case_a,te=te,coef=coef
IDL>print,'te=',te
IDL>print,'coef=',coef
```

Parametric radiative recombination coefficients may have additional adjustment parameters to optimise their fit to higher precision results. Such adjustments are only appropriate for recombination to the ground n-shell and are best handled as a multiplier, c_{rr} and electron temperature gradient shift, de_{rr}. The adjustable form is

$$\alpha_r = c_{rr}\,(I_{n_g}/kT_e)^{de_{rr}} \sum_{l\geq l_g} \alpha^r_{n_g l} + \sum_{n,n>n_g;l,l<n}^{n_t} \alpha^r_{nl} \tag{3.17}$$

The procedure **alf_r_tot.pro** has the capacity for using c_{rr} and de_{rr} parameter in case C. The ADAS Project has in progress high precision radiative recombination coefficient production using the multi-electron, multi-configuration structure and radiative transition code *AUTOSTRUCTURE*. These state-selective (both initial and final state) data are archived in ADAS data format *adf48*. In principle, the case C parameters c_{rr} and de_{rr} will be optimised with respect to these latter data. It will however be some substantial time before the *adf48* data base is complete enough for relevance to heavy element ions. It should finally be noted that radiative recombination is increasingly revealed to be a minor process compared with dielectronic recombination even at very low temperatures. The present procedures are likely to be adequate for most fusion purposes.

3.2.2 Dielectronic recombination

Unlike radiative recombination, dielectronic recombination is strongly dependent on the atomic structure and transitions of the recombining ion, called the *parent ion* or the *core*. Also the effective recombination is dominated by capture to higher n-shells although this dominance becomes less at very high z. In effect the dielectronic recombination contracts towards lower n-shells as the recombining ion charge z increases. It is usual to distinguish situations where the resonant capture part of the composite dielectronic process involves an n-changing core transition, $\Delta n > 0$, or not, $\Delta n = 0$. In the $\Delta n > 0$ case, capture is to lower n-shells and the lowest resonances of this type influence the very low temperature behaviour of dielectronic recombination. In the $\Delta n = 0$ case, capture is into higher n-shells and this type of core transition is in general most influential on dielectronic recombination near ionisation balance in electron excited thermal plasmas such as those of concern in fusion. For dielectronic recombination therefore, the termination of the capture sum to higher n-shells (via an n_t or collisional-radiative population modelling) is a critical matter for modelling finite density plasmas. For recombination of heavy species ions, the precision of dielectronic coefficients depends directly of the quality of the relevant transition energies, resonance capture collision strengths, Auger break-up coefficients and radiative stabilisation transition probabilities. As has been found in earlier chapters, the complexity of heavy element ions limits what can be achieved. We write

$$\alpha^d_{eff} = \sum_{n_c,l_c}\sum_{l'_c}\sum_{n,l:l<n}^{n_t} \alpha^d_{n_c l_c,n_c l'_c;nl} + \sum_{n_c,l_c}\sum_{n'_c:n'_c>n_c,l'_c}\sum_{n,l:l<n}^{n_t} \alpha^d_{n_c l_c,n'_c l'_c;nl} \tag{3.18}$$

distinguishing the transitions $n_c l_c \rightarrow n'_c l'_c$ of the active core electron from the ground state of the recombining ion and the $\Delta n_c = 0$ and $\Delta n_c > 0$ cases. nl is the high orbital into which the free electron is captured. It is helpful to note that $\alpha^d_{n_c l_c,n'_c l'_c;nl}$ takes the form

$$\alpha^d_{n_c l_c,n'_c l'_c;nl} = 4\left[\frac{\pi a_0^2 I_H}{kT_e}\right]^{3/2} \frac{\omega(n'_c l'_c,nl)}{\omega(n_c l_c)} \frac{A^a A^r}{\sum A^a + \sum A^r} e^{-E/kT_e} \tag{3.19}$$

where A^a and A^r represent the combinations of Auger and spontaneous emission coefficients delivering the final stabilised recombined state via the $n'_c l'_c; nl$ resonance configuration , the ω denote statistical weights and E is the energy of the recombining electron for the resonance.

Parametric forms

Dielectronic recombination calculation for plasmas has been hugely influenced by the Burgess general formula (Burgess, 1965). It provides a simple expression for

$$\alpha^d_{n_c l_c,n'_c l'_c;tot} = \sum_{n,l:l<n}^{\infty} \alpha^d_{n_c l_c,n'_c l'_c;nl} \tag{3.20}$$

This is for the total dielectronic recombination for a particular core transition at zero density. It takes as parameters the upward oscillator strength of the core transition $n_c l_c \rightarrow n'_c l'_c$ (only dipole allowed transitions are permitted) and the transition energy. It is available as an IDL procedure **alf_d_bgf** where the acronym *bgf* denotes *Burgess general formula*. This formula can be applied to complex ion recombination providing the relevant core transition and energy data are available. It is to be noted though that the formula specification has a range of recombined ion charge $z_{ion} < 26$ although the formula is freely used beyond this range. The formula has no capability for truncating the n-shell sum at an n_t, however an empirical density dependent reduction factor is often used.

$$\begin{aligned} D &= n_t/(200 + n_t) \text{ for } \Delta n_c = 0 \\ &= 0.0015[(z_1 + 1)n_t]^2/(1 + 0.0015(z_1 + 1)n_t]^2) \text{ for } \Delta n_c > 0 \end{aligned} \tag{3.21}$$

where z_1 is the recombining ion charge and n_t comes from equation 3.12. This was inferred from very early studies of dielectronic recombination for He^+ and Ca^+ and has no generally demonstrated applicability for heavy species (the full *GCR* calculations for light elements bypass this problem). In ADAS, the approach using the *bgf* is called Case A. The underlying FORTRAN subroutine for *bgf* is **xxdrbf.for**. It does not include the above density dependent correction, but the latter can be introduced via keywords in the IDL procedure **alf_d_bgf** as illustrated below.

```
IDL> z0_nuc=2
IDL>z_ion=0
IDL>deij=3.0
IDL>fij=0.416
IDL>z1_ion=z_ion+1
IDL>tev=tev_alf_s,z1_ion,ionpot
IDL>low=11605.4*tev/100
IDL>high=11605.4*tev*100
IDL>te=adas_vector(low=low,high=high,num=21)
IDL>alf_d_bgf,z0_nuc,z_ion,deij=deij,fij=fij,te=te,coef_tot=coef_tot
IDL>print,'te=',te
IDL>print,'coef_tot',coef_tot
IDL>dens=1.0e14
IDL>delta_nc=1
IDL>alf_d_bgf,z0_nuc,z_ion,deij=deij,fij=fij,te=te,dens=dens,delta_nc=delta_nc,coef_tot=coef_tot
IDL>print,'dens',dens
IDL>print,'coef_tot',coef_tot
```

Some of the immediate deficiences of *bgf* are resolved by the code called the *Burgess General Program* or *bgp* for short. This evaluates the separate contributions of capture to nl-shells and the partial sums over l and then n. Introduction of an n_t truncation is possible in *bgp*. A more sophisticated improvement is also possible. *bgp* calculates dipole *nl*-selective resonance capture in the Bethe approximation via a Correspondence Principle argument. This relates the Auger rate coefficients to Bethe approximation excitation collision strengths at threshold. Low partial wave corrections to the Bethe approximation values are the key to enhanced accuracy. *bgf* is a functional fit to extensive *bgp* calculations at zero density. The latter used a fixed set of Bethe correction factors based on cross-section data available at the time. An improvement is to use sets of correction factors for various types of parent transitions. These were obtained for medium weight elements by (Summers *et al* , 1987). The IDL procedure **alf_d_bgp** provides the *bgp* approximation at the command line for a single parent transition. It provides the total zero-density recombination coefficient at a set of electron temperatures. Optionally a density dependent reduction may be imposed based on the simple n_t cut-off above. The procedure also can return the partial n-shell coefficients on a representative n-shell set and force the use of a particular set of Bethe correction factors. There are several optional parameters and extensive use of deafult values. The underlying FORTRAN subroutines are **xxdrbp.for** and **gxdrbp.for**. The former gives the recombination to an n-shell and its l-subshells. The latter provides the total, density corrections and results for representative n-shells. It is illustrated below and corresponds to the ADAS Case B.

```
IDL> z0_nuc=2
IDL>z_ion=0
IDL>ep=3.0
IDL>fp=0.416
IDL>np=2
IDL>lp=1
IDL>ng=1
IDL>lg=0
IDL>z1_ion=z_ion+1
IDL>tev=tev_alf_s,z1_ion,ionpot
IDL>low=11605.4*tev/100
IDL>high=11605.4*tev*100
IDL>te=adas_vector(low=low,high=high,num=21)
IDL>alf_d_bgp,z0_nuc=,z0_nuc,z_ion=z_ion,ep=ep,fp=fp,np=np,lp=lp,ng=ng,lg=lg,te=te,coef_tot=coef_tot
IDL>print,'coef_tot',coef_tot
IDL>cor=[0.05,0.3,0.7,0.9]
IDL>nmin=2
IDL>dens=1.0e10
IDL>alf_d_bgp,z0_nuc=,z0_nuc,z_ion=z_ion,ep=ep,fp=fp,cor=cor,nmin=nmin,te=te,dens=dens,coef_tot=coef_tot
IDL>print,'te=',te
IDL>print,'dens',dens
IDL>print,'cor',cor
IDL>print,'coef_tot=',coef_tot
IDL>nrep=[2,3,4,5,6,7,8,9,10,12,15,20,30,40,50,60,70,100,150,200,250,300,400,500,600,700,800,900,1000]
IDL>alf_d_bgp,z0_nuc=,z0_nuc,z_ion=z_ion,ep=ep,fp=fp,cor=cor,nrep=nnrep,te=te,coef_n=coe_n,coef_tot=coef_tot
IDL>print,'nrep',nrep
IDL>print,'coef_n(nrep(10),te)',coef_n(10,*)
```

Alternate Auger channels open extra loss channels competing with stabilisation and can be included in the *bgp* formulation to some extent. To a good approximation, it can be assumed that once an alternate channel is open then there is unit branching ratio in its favour. Thus an n-shell cut-off, n_{aa} may be introduced based only on energetics. In practice, the lower of n_t and n_{aa} is applied. All necessary data for the above two aspects can be obtained from ADAS *adf04* files.

The ADAS Project has had in progress, for a number of years, high precision dielectronic recombination coefficient production using the multi-electron, multi-configuration structure and radiative transition code *AUTOSTRUCTURE* - the 'DR Project'(see *DR - paper I*). These state-selective (both initial and final state) data are archived in ADAS data format *adf09*. Project results have to-date been completed up to the Mg-like iso-electronic sequence and are used in the *GCR* models of light elements. It is expected that these calculations will continue to progress towards heavy element ions, but it is evident that the scale of the calculations and the data storage requirement of the dielectronic recombination database are becoming unbalanced. A modified strategy is in progress to solve these issues which is usefully described here.

The precision of the DR Project calculations, and its divergences from *gbp*, arise from the inclusion of non-dipole as well as dipole excitations from the parent ground state in the resonance formation, the exact inclusion of alternative Auger and radiative branching channels for resonance decay, stabilisation via outer electron transition into the core and the specificity of the collision data used. Additionally, detailed attention is given to the lowest lying resonances which determine the low temperature behaviour. In the latter context DR Project calculations for an ion are generally resolved into parts with titles such as '2 to 3, capture to 3' or '2 to 3, capture to n' meaning core transitions from $n_c = 2$ to $n_{c'} = 3$ with captured electron entering principal quantum shell $n = 3$ in the first case and with the captured electron entering any principal quantum shell $n > 3$ in the second. These are written in shorthand as *2-3,3* and *2-3,n* respectively. The first case is the sort which spans the difficult low lying resonances, but are quite limited in number. The second case generates the infinite series of resonances and the very large data sets for heavy species ions. This latter case is amenable to economised calculation by an extension of *gbp* which we call *bbgp*. It is the new *bbgp* which

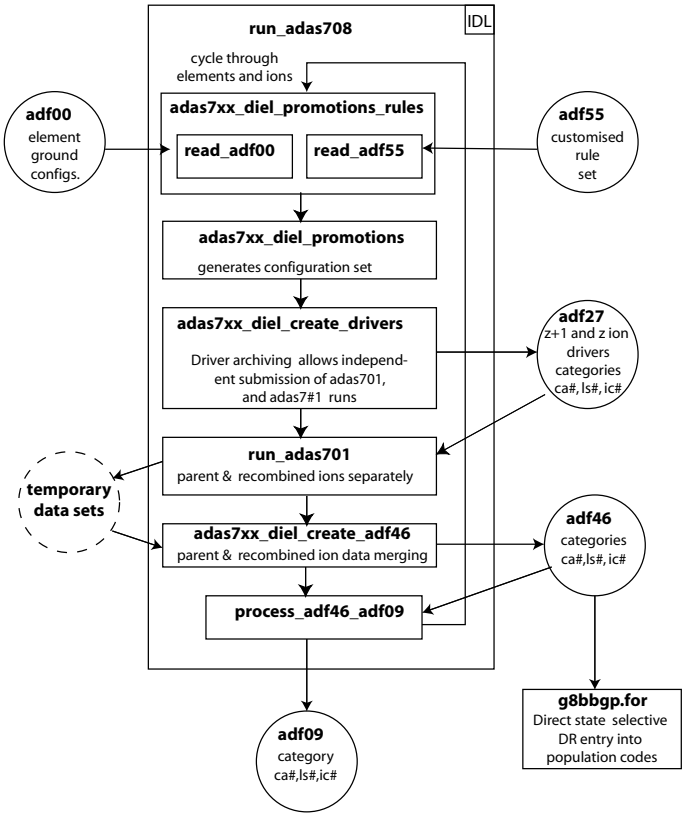

Figure 3.4: Organisation and disposition of data sets used and created in offline execution of ADAS708

we introduce here and which and allows us to move the baseline data for dielectronic recombination to Case C.

$$cor_l = \sum_{l'} \Omega((S_p L'_p J'_p) k' l', (S_p L_p J_p) k|_{=0} l) \Big/ \sum_{l'} \Omega^{\rm Bethe}((S_p L'_p J'_p) k' l', (S_p L_p J_p) k|_{=0} l) \,. \tag{3.22}$$

In principle, the cases B and C can have adjustable parameters c_{dr} and de_{dr} which may be optimised with respect to available high quality data, from the DR project, iso-electronic sequence by iso-electronic sequence. These parameters are introduced most effectively as a scaling and an effective transition energy shift. Up to two parent transition groups corresponding to $\Delta n_c = 0$ and $\Delta n_c > 0$ may be used, as indicated in the equation below. The factor $a = 1.0 + 0.015[z_1^3/(z_1+1)^2]$, which compensates for the energy shift of resonance energies from associated core transition energies, is helpful and is taken from the *GF* specification.

Calculation of promotion rules for DR

Calculation of the DR rates utilising the autostructure code requires that suitable input files are generated. A set of promotion rules has been devised to determine the correct configurations to include. These rules are stored in an ADF55 file, and are identical to those in an ADF54 file, with the addition of the spectator electron n-shell and the maximum l-shell to be considered for capture of the electron.

The addition of the spectator electron increases the calculation size considerably compared to the structure calculations considered in section 2.2. Therefore the number of configurations may need to be reduced. It is therefore once again worth operating the promotional rules code, with a smaller target size. Since the Autostructure code operates in LS coupling, the taraget calculation size is best optimised for the number of terms, not the number of levels as before. This is done by altering the last parameter from a 1 to a 2, and changing the number of terms appropriately. The following commands will run this optimisation for a target size of 200 terms for h-like to li-like magnesium:

>run_opt_808_offline.sh mg 1 3 /home/<user>/work 12 magnesium 1 200
<ADASDIR>/adas/adf54/promotion_rules_medium_adf54.dat 2

Once this ADF54 has been generated, the updated rules can be copied to an ADF55 file with the command:
IDL> adas8xx_recom_merge_adf54_adf55, a55file='my_adf55_file.dat',$
a54file='my_adf54_file.dat'

3.3 Finite density heavy species collisional-radiative coefficients

The parameters required to calculate the Case A and Case B approximations to $S^{z\rightarrow z+1}$ and $\alpha^{z+1\rightarrow z}$ are conveniently stacked together for $z = 0, \cdots, z_0 - 1$ in ADAS data format *adf03*. As has been described earlier, these parameters may be extracted from the set of *adf04* data sets for all the ionisation stages of an element. The code ADAS407 performs this process. Then the code ADAS408 generates the tables of the coefficients, essentially by running the appropriate procedures of sub-sections 3.1.1, 3.2.1 and 3.2.2, as required for application in ADAS format *adf11*. These are collisional-radiative coefficients with a limited density dependence arising from the n_t cut-off. This approximation means that the coefficients are valid at low to moderate scaled electron densities spanning most of the tokamak relevant regime (excluding the lowest few ionisation stages and super-high densities. These steps may be executed at the IDL command line as

```
IDL>te=adas_vector(low=low,high=high,num=21)
IDL>run_adas407
IDL>print,'te=',te
IDL>print,'coef=',coef
IDL>run_adas408
IDL>print,'te=',te
IDL>print,'coef=',coef
```

As described in sub-sections 3.1.1, 3.2.1 and 3.2.2, Case B and Case C methodologies have implicit n-shell resolution of the radiative recombination processes. So a population model within the *bundle-n* approximation is possible. The ADAS Project has a substantial capability for this. The code ADAS316 is particularly suited to the baseline heavy species situation. It handles an effectively infinite number of n-shells, evaluating representative n-shell populations and the collisional-radiative recombination and ionisation coefficients. The latter include stepwise processes with matched forward and reverse processes. Thus stepwise ionisation and three-body recombination are included in its evaluation of $S^{z\rightarrow z+1}$ and $\alpha^{z+1\rightarrow z}$. The calculations are valid to very high densities converging correctly on the local thermodynamic equilibrium limit. We have provided a capbility for execution of ADAS316 at the IDL command line, enabled to write directly the *adf11* data files. This is illustrated below.

```
IDL>te=adas_vector(low=low,high=high,num=21)
IDL>alf_d_bgf,z0_nuc,z_ion,case=case_a,te=te,coef=coef
IDL>print,'te=',te
IDL>print,'coef=',coef
IDL>alf_d_bgp,z0_nuc,z_ion,case=case_a,te=te,coef=coef
IDL>print,'te=',te
IDL>print,'coef=',coef
```

Chapter 4

Superstages and flexible partitioning

In the generalised-collisional-radiative picture, an element in a plasma is described by the abundances of all the metastables of every ionisation stage, by the effective recombination and ionisation coefficients which link them together and by the emission coefficients which are quasi-static with respect to and driven by these metastables. The complete set of populations, which are then in principle tracked in dynamic transport modelling, is large. However, not all populations are of equal importance and so grouping of populations may be appropriate. This is called a condensation and it reduces the problem to tracking the group populations with their equivalent effective recombination and ionisation coefficients and emission coefficients. The specification of a grouping is called a *partition*. More precisely, introduce a parent partition with n_p members indexed as

$$\{i : i = 0, \cdots, n_p\}_p \tag{4.1}$$

then we can define a child partition with n_c members, indexed as

$$\{j : j = 0, \cdots, n_c\}_c \tag{4.2}$$

by a range vector

$$\{r_j : j = 0, \cdots, n_c\}_c \tag{4.3}$$

so that

$$\{j\}_c \quad \equiv \quad \{R_{j-1}, \cdots, R_j - 1\}_p \tag{4.4}$$

where $R_j = \sum_{k=0}^{j} r_j$ and $R_{-1} = 0$. The r_j determine the contiguous ranges in the parent partition index to be grouped into the child partition members j. For example a parent partition with six members may be grouped into two child partition members with $r_1 = 2$ and $r_2 = 4$ so that $\{1\}_c = \{1, 2\}_p$ and $\{2\}_c = \{3, 4, 5, 6\}_p$. We can envisage a particular partition having parent and grandparent partitions and so on in *layers* back to a *root partition*. This is the basic idea, but it can be made more complete and consistent with earlier work in both the *CR* and *GCR* pictures.

Consider the metastables of carbon in the *GCR* picture, as shown schematically in figure 4.1a. The complete set of thirteen metastables (including the bare nucleus) comprise the members of a partition. If the metastables of each ionisation stage are grouped together, then the child partition is composed of members which are the ionisation stages. Then further grandchild partitions can be formed as illustrated. There are really two starting points for partitioning, called the #00 or #01 *root partition layers* depending on whether we are in the *GCR* picture of resolved metastables or the unresolved *CR* picture. For computation, a partition specified by $\{j, r_j : j = 0, \cdots, n\}$ is conveniently written as a string, so that the two root partitions layers for carbon appear respectively as

$$\begin{aligned} //\#00// \quad & p00/\ 00/p01/\ 01/p02/\ 02/p03/\ 03/p04/\ 04/ \\ & p05/\ 05/p06/\ 06/p07/\ 07/p08/\ 08/p09/\ 09/ \\ & p10/\ 10/p11/\ 11/p12/\ 12/p13\ 13/ \end{aligned} \tag{4.5}$$

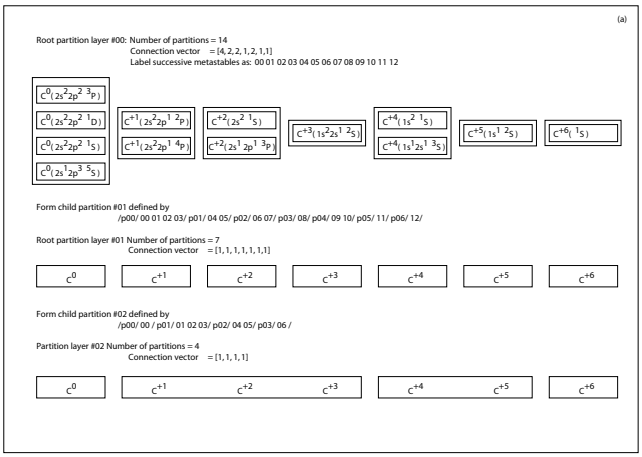

Figure 4.1: (a) Metastables of carbon in the resolved *GCR* picture. The child partition grouping to form the stage to stage picture is shown as the boxes enclosing the metastable sets. The connection vector specifies the metastables which are close-coupled. It corresponds in this particular illustration to the child partition grouping. (b) The general partitioning picture with independent child partition groupings and parent connection vector. Note that the child connection vector is determined from the child partition definition and the parent connection vector.

and

$$\begin{array}{ll} //\#01// & p00/\ 00/p01/\ 01/p02/\ 02/p03/\ 03/p04/\ 04/ \\ & p05/\ 05/p06/\ 06/ \end{array} \tag{4.6}$$

The child #01 stage-to-stage partition of carbon for computation from the #00 root partition is

$$\begin{array}{ll} //\#01// & p00/\ 00\ 01\ 02\ 03/p01/\ 04\ 05/p02/\ 06\ 07/ \\ & p03/\ 08/p04/\ 09\ 10/p5/\ 11/p6/\ 12/ \end{array} \tag{4.7}$$

It should be noted that the metastables are not all on the same footing. Unlike ionisation stages which are each (usually) only coupled to the two adjacent stages up and down, the metastables are in groups which are close-coupled (for example the first four and the next two and so on). This close coupled grouping is specified by the *connection vector*, a name which will be familiar to those who have worked with ADAS *GCR* data. The connection vector for carbon in the #00 root partition is $\{4, 2, 2, 1, 2, 1, 1\}$. The connection vector is similar to a range vector and the range vector for grouping into the #01 stage-to-stage partition from the #00 root partition is the connection vector. Introduce a new name *superstage* for the members of a partition which are close coupled as specified by the connection vector. In the special case of the #01 partition of carbon, the superstages are simply ordinary stages. The connection vector and range vector of a child partition are in the general case independent items. So a child partition can be formed from the #00 root partition which subdivides connected metastables into different child partition members and leaves a non-trivial connection vector for the child partition. This general case is illustrated in figure 4.1b. With light elements, in usual plasma conditions, ionisation is a single electron loss process from ionisation stage $\mathcal{A}^{+z}$ to $\mathcal{A}^{+z+1}$. For heavy element ions, multiple electron loss through shake-down and shake-off is more likely. This would result in non-trivial connection vectors and superstages. The heavy element baseline has the the root partition layer #01 and in child partitions, the connection vector is trivially all 'ones'.

Consider then the evolution of populations of members of a partition of an element in a plasma. For an element $\mathcal{A}$ of nuclear charge z_0, in the unresolved picture, the populations of the superstages of the #01 partition are denoted by

$$N^{[\#01](i)} : i = 1, \cdots I^{[\#01]} \tag{4.8}$$

and their time dependence is given by the equations

$$\begin{aligned} \frac{dN^{[\#01](i)}}{dt} &= N_e S_{cr}^{[\#01](i-1\rightarrow i)} N^{[\#01](i-1)} \\ &\quad -N_e (S_{cr}^{[\#01](i\rightarrow i+1)} + \alpha_{cr}^{[\#01](i\rightarrow i-1)}) N^{[\#01](i)} \\ &\quad +N_e \alpha_{cr}^{[\#01](i+1\rightarrow i)} N^{[\#01](i+1)} \end{aligned} \tag{4.9}$$

The coefficients are the #01 partition collisional-radiative coefficients, corresponding to the usual stage to stage coefficients $S_{cr}^{(z-1\rightarrow z)}$ and $\alpha_{cr}^{(z\rightarrow z-1)}$ since the superstage index i is the same as the charge state z. Consider now the child partition layer '#02'. Without loss of generality suppose that the members of layer #01 between index values i_0 and i_1 map into member j of #02. Then

$$N^{[\#02](j)} = \sum_{i=i_0}^{i_1} N^{[\#01](i)} \tag{4.10}$$

and summing the time dependent equations

$$\begin{aligned} \frac{dN^{[\#02](j)}}{dt} &= N_e S_{cr}^{[\#01](i_0-1\rightarrow i_0)} N^{[\#01](i_0-1)} \\ &\quad -N_e \alpha_{cr}^{[\#01](i_0\rightarrow i_0-1)} N^{[\#01](i_0)} \\ &\quad -N_e S_{cr}^{[\#01](i_1\rightarrow i_1+1)} N^{[\#01](i_1)} \\ &\quad +N_e \alpha_{cr}^{[\#01](i_1+1\rightarrow i_1)} N^{[\#01](i_1+1)} \end{aligned} \tag{4.11}$$

Impose a quasi-static equilibrium for the #01 partition members populations i_0 to i_1 so that

$$\begin{aligned} N^{[\#01](i_0)}\Big|_{eq} &= (\alpha_{cr}^{[\#01](i_0+1\to i_0)}/S_{cr}^{[\#01](i_0\to i_0+1)})N^{[\#01](i_0+1)}\Big|_{eq} \\ N^{[\#01](i_0+1)}\Big|_{eq} &= (\alpha_{cr}^{[\#01](i_0+2\to i_0+1)}/S_{cr}^{[\#01](i_0+1\to i_0+2)})N^{[\#01](i_0+2)}\Big|_{eq} \\ &\cdots \\ N^{[\#01](i_1-1)}\Big|_{eq} &= (\alpha_{cr}^{[\#01](i_1\to i_1-1)}/S_{cr}^{[\#01](i_1-1\to i_i)})N^{[\#01](i_1)}\Big|_{eq} \end{aligned} \tag{4.12}$$

subject to the normalisation

$$N^{[\#02](j)} = \sum_{i=i_0}^{i_1} N^{[\#01](i)}\Big|_{eq} \tag{4.13}$$

Then finally define the effective recombination and ionisation coefficients for the #02 partition as

$$\begin{aligned} \alpha_{cr}^{[\#02](j\to j-1)} &= \alpha_{cr}^{[\#01](i_0\to i_0-1)}\left(N^{[\#01](i_0)}/N^{[\#02](j)}\right)\Big|_{eq} \\ S_{cr}^{[\#02](j\to j+1)} &= S_{cr}^{[\#01](i_1\to i_1+1)}\left(N^{[\#01](i_1)}/N^{[\#02](j)}\right)\Big|_{eq} \end{aligned} \tag{4.14}$$

When there is no confusion, the $^{[\#<nn>]}$ partition layer label in the superscripts can be omitted leaving the familiar older notation.

For the general case, there may be more than one partition member in a superstage and the population are written as $N_\rho^{[\#<nn>](i)}$ where i is the superstage index and ρ the member index within the superstage. The vector of populations of members of a superstage, is written in script character $\mathcal{N}^{[\#<nn>](i)}$. To achieve a parent to child condensation in the general case, it is helpful to work with the matrix of equilibrium fractional populations in the parent partition layer, renormalised separately to fractional populations within each child partition member as (without loss of generality suppose the parent partition is #00)

$$\mathbf{F} = \begin{bmatrix} [\mathcal{F}_{[\#00]_i\in[\#01]_0,[\#01]_0}] & 0 & . & 0 \\ 0 & [\mathcal{F}_{[\#00]_i\in[\#01]_1,[\#01]_1}] & . & 0 \\ . & . & . & . \\ 0 & 0 & . & [\mathcal{F}_{[\#00]_i\in[\#01]_{N_c},[\#01]_{N_c}}] \end{bmatrix} \tag{4.15}$$

where the $[\mathcal{F}_{[\#00]_i\in[\#01]_j,[\#01]_j}]$ represent column vectors with elements

$$\begin{aligned} \mathcal{F}_{[\#00]_i\in[\#01]_j,[\#01]_j} &= N^{[\#00](i)}/N^{[\#01](j)} \\ &= N^{[\#00](i)}/\sum_{k\in[\#01]_j} N^{[\#00](k)} \end{aligned} \tag{4.16}$$

and the similar matrix

$$\mathbf{I} = \begin{bmatrix} [\mathcal{I}_{[\#00]_i\in[\#01]_0,[\#01]_0}] & 0 & . & 0 \\ 0 & [\mathcal{I}_{[\#00]_i\in[\#01]_1,[\#01]_1}] & . & 0 \\ . & . & . & . \\ 0 & 0 & . & [\mathcal{I}_{[\#00]_i\in[\#01]_{N_c},[\#01]_{N_c}}] \end{bmatrix} \tag{4.17}$$

where the $[\mathcal{I}_{[\#00]_i\in[\#01]_j,[\#01]_j}]$ represent column vectors with unit elements

$$\mathcal{I}_{[\#00]_i\in[\#01]_j,[\#01]_j} = 1 \tag{4.18}$$

Also assemble the collisional-radiative matrix formed from submatrices of dimensionalities those of the parent superstage connection vector values as

$$\mathbf{C} = \begin{bmatrix} \mathcal{C}^{[\#00](0\to 0)} & N_e\,\mathcal{R}^{[\#00](1\to 0)} & . & . \\ N_e\mathcal{S}^{[\#00](0\to 1)} & \mathcal{C}^{[\#00](1\to 1)} & . & . \\ . & N_e\,\mathcal{S}^{([\#00]1\to 2)} & . & . \\ . & . & . & . \end{bmatrix} \tag{4.19}$$

Note that the on-diagonal elements of the collisional-radiative matrix must be formed correctly so that the column sums are zero. Then the child partition collisional-radiative matrix is $\mathbf{D} = \mathbf{I}^{\mathrm{T}}\mathbf{CF}$. This must be written in the submatrix structure of the child superstages following the child connection vector element values as

$$\mathbf{D} = \begin{bmatrix} \mathcal{C}^{[\#01](0\rightarrow 0)} & N_e\,\mathcal{R}^{[\#01](1\rightarrow 0)} & . & . \\ N_e\mathcal{S}^{[\#00](0\rightarrow 1)} & \mathcal{C}^{[\#00](1\rightarrow 1)} & . & . \\ . & N_e\,\mathcal{S}^{[\#00]1\rightarrow 2)} & . & . \\ . & . & . & . \end{bmatrix} \tag{4.20}$$

from which the individual collisional-radiative coefficients of the child partition layer may be 'unpicked'.

The issue of thermal charge exchange recombination has not been addressed. The condensation process is non-linear for linear combinations of 'acd' and 'ccd' *adf11* data. In the present work the condensation is carried out using only the free-electron electron driven recombination, that is $\mathcal{R}$ excludes charge exchange recombination as does the quasi-static ionisation balance of matrix **F**. Thus the child 'ccd' data must be obtained by forming separately a **CX** matrix as for **C** above but including only the neutral hydrogen driven charge exchange sub-matrices $\mathcal{R}^{CX}$. The condensation and unpicking is as for **C** above. In the light of this approximation, if there is strong charge exchange to a particular ionisation stage, then this stage should be isolated as its own superstage in condensations. In terms of these new coefficients, which include recombination, ionisation (and cross-coupling in the general case) , the usual form of the time dependent equations for the #02 superstage populations (and the superstage partition members in the general case) are obtained. In practice, superstage condensation (bundling) is influenced by the shell structure of the element. There are two situations of interest at this time. Firstly there is aggressive condensation (bundling) aimed at enabling heavy elements to be handled with the same ecomomy as light elements in sophisticated 2-d or 3-d transport codes. Secondly, there is condensation related to spectroscopy and the occurrence of quasi-continuum. It is in these contexts that the additional classes in the *adf11* data format and those in the *adf15* and *adf40* formats are determined for the child partition in the following two sub-sections.

4.1 The natural partition and spectroscopy

Ions with closed shell configurations have an extended region of existence (in electron temperature) in a plasma. They and adjacent alkali-like ions are significant radiators of a resolvable line spectrum of diagnostic value. Sets of ions associated with a partially filled shell, particularly for heavy species, give a complex overlapped spectrum which is difficult to identify and resolve. They are usefully grouped. A map of the fractional variation of ionisation potential $2(I_{z+1} - I_z)/(I_{z+1} + I_z)$ between successive ions of every element, as shown in figure 4.2, highlights the shell structure. We select ions corresponding to peaks and their immediate neighbours as individuals in a partition and group the others. Setting a fraction of the a running mean as a variance for stage individualsing allows automatic partitioning. We call this the *natural partition*.

It is useful to be able to preview the natural partition and its approximate implications for the temperature distribution of superstages before executing the substantive generation of a complete ADAS data for a new child partition layer. An IDL procedure

preview_natural_partition.pro

accomplishes this. At the IDL command line type

IDL>*z0=73*
IDL>*sigma = 1.5*
IDL>*plot_no = 4*
IDL>*preview_natural_partition,z0=z0,sigma=sigma*

The textlines of the parent and child partition layers, as required for driver scripts, are written to the screen and the figure is displayed. There are keywords to send the textlines and graphs to files. Use the form

IDL>*preview_natural_partition,z0=z0,sigma=sigma,/postscript,/save_partition"*

The files names are predetermined and placed in the IDL launch directory as

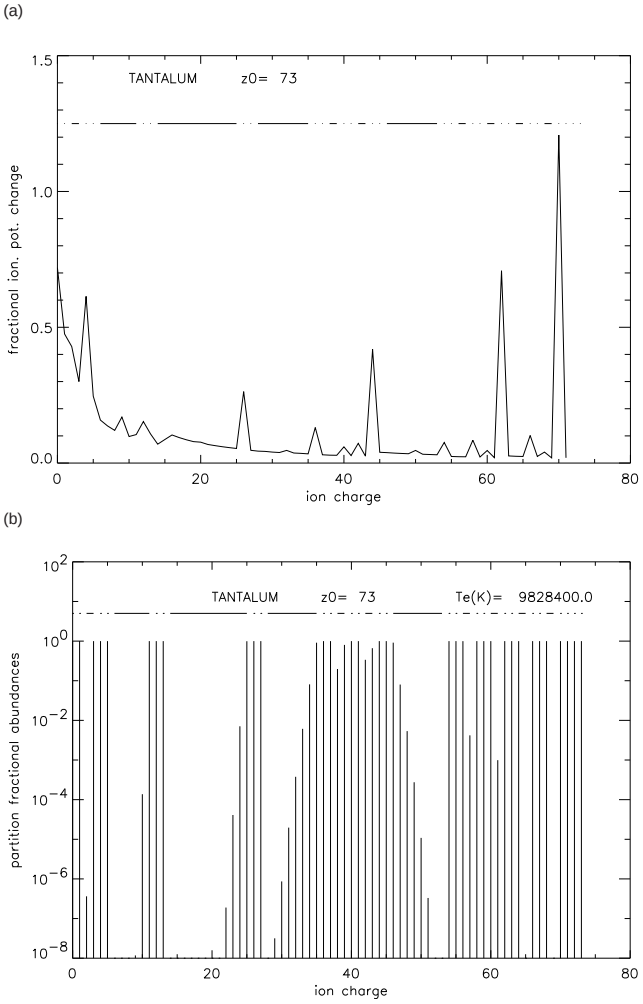

Figure 4.2: (a) Fractional variation of ionisation potential for the ions of tantalum. The peaks indicate the quantum shell boundaries. The natural partition criterion is set at the 3σ level and the individualised stages and bundles are indicated by dots and bars respectively above the graph. (b) Quasi-equilibrium fractions for each ionisation stage within a bundle with respect to the whole bundle.

preview_fig<fig. no.>_<elem. symb.>.ps
preview_partition_stack_<elem. symb.>.txt

Plots available are 1: ionisation potential variation, 2: quasi-equilibrium fractions within a superstage, 3: stage-to-stage ionisation balance, 4: superstage ionisation balance. The output temperature used for fractional abundances within a superstage for the plot of type 2 is representative.

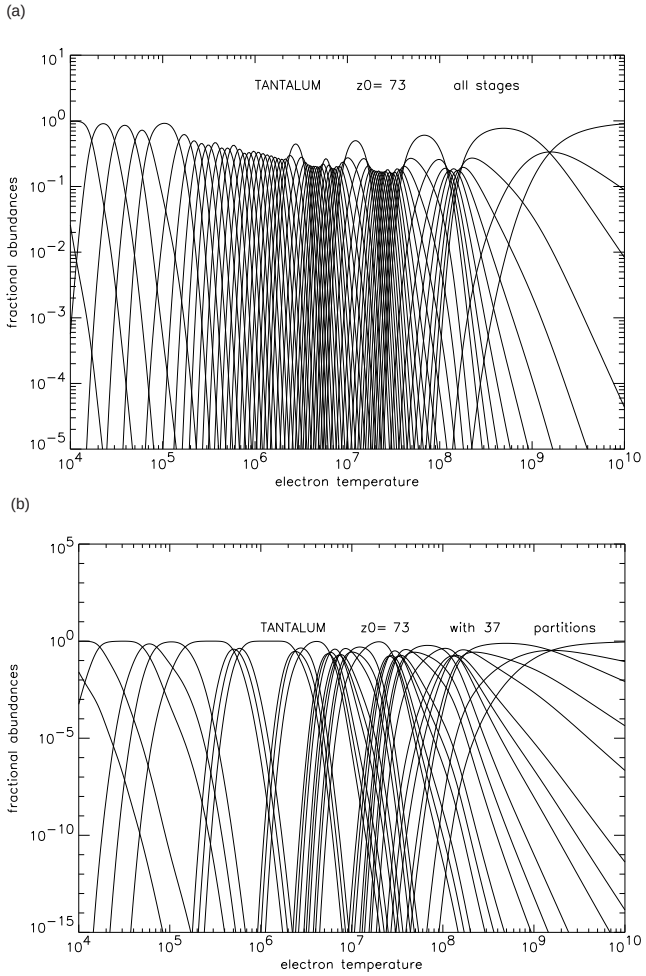

Figure 4.3: Ionisation balance for tantalum vs electron temperature. (a) Equilibrium fractional abundances with unresolved individualised stages corresponding to the //#01/ partition. (b) Equilibrium fractional abundances for ionisation stage bundles in the natural partition of figure 4.2a.

For the spectral interval [0, 1], the envelope feature photon emissivity function vector is

$$\mathcal{FGJN}^{[\#02][0,1](j)} = \sum_{i_0}^{i_1} \mathcal{FGJN}^{[\#01][0,1](i)} \left(N^{[\#01](i)} / N^{[\#02](j)} \right) \Big|_{eq} \tag{4.21}$$

In practice, a number of conventions have been introduced. The baseline modelling, in so far as it does not distinguish metastables of ionisation stages is called the standard unresolved case. Conventionally , the root partition of individual states is called the #01 partition. Also, partition members are indexed starting at zero. Thus the ion charge and the partition index are equivalent for the standard unresolved #01 partition and

$$\mathcal{FGJN}^{[\#01][0,1](z)} = \mathcal{FPEC}^{[0,1](z)} \tag{4.22}$$

Commonly, the envelope feature emission function for an element in full ionisation balance is required. Starting from

the standard unresolved root partition, the required function is

$$\mathcal{FGTN}^{[\#02][0,1](0)} = \sum_{i_0}^{i_1} \mathcal{FGTN}^{[\#01][0,1](z)} \left(N^{(z)} / N^{[\#02](0)} \right) \Big|_{eq} \tag{4.23}$$

4.2 Superstage condensation and plasma transport models

As discussed earlier, a main advantage of the condensation is in economising complex transport calculations without severe loss of accuracy.

From the equilibrium stage population solution, the radiated power function for the child partition is $P_{tot}^{[\#02](j)}$ and is calculated as

$$\begin{aligned} P_{tot}^{[\#02](j)} &= \sum_{i_0}^{i_1} P_{tot}^{[\#01](i)} \left(N^{[\#01](i)} / N^{[\#02](j)} \right) \Big|_{eq} \\ &= \sum_{i_0}^{i_1} \left[P_{LT}^{[\#01](i)} + P_{RB}^{[\#01](i)} \right] \left(N^{[\#01](i)} / N^{[\#02](j)} \right) \Big|_{eq} \end{aligned} \tag{4.24}$$

with separate *radiated power function* contributions arising from low level line power and the recombination-bremsstrahlung-cascade power.

However there are some new issues which must be addressed first before superstages are compatible with transport models. A superstage, which is a composite of several ionisation stages has a superstage charge which depends on electron temperature and electron density. It is a collisional-radiative quantity. In fact fluid transport models make use of the ion charge, squared ion charge and ionisation potential in addition to familar quantities such effective ionisation, recombination, radiated power and electron energy loss coefficients. The *adf11* classes must be extended with these extra quantities, which in superstages are all collisional-radiative quantities. In ADAS, these are given the mnemonics *zcd*, *ycd* and *ecd*. ADAS codes and data structures have been adjusted accordingly and there is a revised specification of *adf11*. The definition and computation of *zcd* and *ycd* are straightforward.

$$\begin{aligned} z^{[\#02]}(j)_{cr} &= \sum_{i \in p_j^{[\#02]}} z^{[\#01]}(i)_{cr} \left(N^{[\#01](i)} / N^{[\#02](j)} \right) \Big|_{eq} \\ (z^2)^{[\#02]}(j)_{cr} &= \sum_{i \in p_j^{[\#02]}} (z^2)^{[\#01]}(i)_{cr} \left(N^{[\#01](i)} / N^{[\#02](j)} \right) \Big|_{eq} \end{aligned} \tag{4.25}$$

ecd is more subtle. Consider a lowest superstage which is a condensation of low ionisation stages in addition to the first. Then from the point of view of energy conservation, the superstage has a birth energy associated with its appearance in the plasma. In the usual (simplified) ionisation stage picture, a neutral has zero birth energy, which is implicit in the computer codes. For consistency then, there is a 0^{th} block in the **adf11/ecd** dataset which is the birth energy of the lowest superstage. The #01 root partition has all zeros in this block. In fact this problem is already present in the *GCR* picture, since the energy of metastables of the lowest ionisation stage are usually ignored. A properly formed *ecd* in the *GCR* picture should contain excitation energies from the lowest metatable (the ground state) to higher metastables of the same stage as well as ionisation potentials to the metastables of the next stage. These data are available for selected elements within ADAS under data format *adf00* in data sets of the form

<el.symb.>_ls.dat

<el.symb.>_ic.dat

for *ls* and *ic* cases in addition to the usual unresolved form

<el.symb.>.dat

The FORTRAN subroutine **xxdata_00.for** returns extra information on the metastables, including their configurations. The IDL procedure can also acquire these data if the keyword */ls* or */ic* is appended to the call.

Form an augmented excitation/ionisation energy matrix as

$$\mathbf{E} = \begin{bmatrix} 0 & \mathcal{E}^{[\#00](-1\to0)} & . & . & . \\ . & \mathcal{E}^{[\#00](0\to0)} & \mathcal{E}^{[\#00](0\to1)} & . & . \\ . & . & \mathcal{E}^{[\#00](1\to1)} & \mathcal{E}^{[\#00](1\to2)} & . \\ . & . & . & \mathcal{E}^{[\#00](2\to2)} & . \\ & & & & \\ . & . & . & . & . \end{bmatrix} \tag{4.26}$$

The submatrix $\mathcal{E}^{[\#00](-1\to0)}$ contains the initial formation energies of the members of the superstage 0 in the plasma. So if connection vector commences $\{3, 2 \cdots\}$ then

$$\mathcal{E}^{[\#00](-1\to0)} = \begin{bmatrix} \mathcal{E}^{[\#00](-1\to0)}_{1\to1} & \mathcal{E}^{[\#00](-1\to0)}_{1\to2} & \mathcal{E}^{[\#00](-1\to0)}_{1\to3} \end{bmatrix} \tag{4.27}$$

and

$$\mathcal{E}^{[\#00](0\to0)} = \begin{bmatrix} \mathcal{E}^{[\#00](0\to0)}_{1\to1} & \mathcal{E}^{[\#00](0\to0)}_{1\to2} & \mathcal{E}^{[\#00](0\to0)}_{1\to3} \\ \mathcal{E}^{[\#00](0\to0)}_{2\to1} & \mathcal{E}^{[\#00](0\to0)}_{2\to2} & \mathcal{E}^{[\#00](0\to0)}_{2\to3} \\ \mathcal{E}^{[\#00](0\to0)}_{3\to1} & \mathcal{E}^{[\#00](0\to0)}_{3\to2} & \mathcal{E}^{[\#00](0\to0)}_{3\to3} \end{bmatrix} \tag{4.28}$$

contains excitation energies between members of the superstage. The elements $\mathcal{E}^{[\#00](0\to0)}_{\rho\to\rho}$ are nomally zero, the elements $\mathcal{E}^{[\#00](0\to0)}_{1\to\sigma}$ with $\sigma > 1$ are excitation energies which come from the tabulations in resolved *adf00* files of type *ls* or *ic* and all other elements of the matrix may be filled in, noting that $\mathcal{E}^{[\#00](0\to0)}_{\sigma\to\rho} = -\mathcal{E}^{[\#00](0\to0)}_{\rho\to\sigma}$. The matrix

$$\mathcal{E}^{[\#00](0\to1)} = \begin{bmatrix} \mathcal{E}^{[\#00](0\to1)}_{1\to1} & \mathcal{E}^{[\#00](0\to1)}_{1\to2} \\ \mathcal{E}^{[\#00](0\to1)}_{2\to1} & \mathcal{E}^{[\#00](0\to1)}_{2\to2} \\ \mathcal{E}^{[\#00](0\to1)}_{3\to1} & \mathcal{E}^{[\#00](0\to1)}_{3\to2} \end{bmatrix} \tag{4.29}$$

contains ionisation energies between members of adjacent superstages. The element $\mathcal{E}^{[\#00](0\to1)}_{1\to1}$ comes from the tabulations in resolved *adf00* files of type *ls* or *ic* and all other elements of the matrix may be filled in once the matrices $\mathcal{E}^{[\#00](0\to0)}$ and $\mathcal{E}^{[\#00](1\to1)}$ have been determined. If the submatrices are filled in this manner so that they span the range of the child #01 connection vector then the production of the child excitation/ionisation energy matrix is straightforward. Prepare an augmented **G** fractional abundance matrix as

$$\mathbf{G} = \begin{bmatrix} 1 & 0 \\ 0 & \mathbf{F} \end{bmatrix} \tag{4.30}$$

Then the child matrix is $\mathbf{G}^{\mathrm{T}}\mathbf{E}\mathbf{G}$ which may be unpicked as before to form the compact child *ecd* file.

As the historical ADAS *adf11* database does not contain *zcd*, *ycd* and *ecd*, a PERL script is available to create them as

/home/adas/offline_adas/adas4#1/scripts/generate_adf11_classes_10-12.pl

By default this scans the **/home/<user>/adas/adf11** subdirectory for the various year numbers (in fact it looks for *plt* classes since these are most suitable as templates for the new classes) and creates matching *zcd*, *ycd* and *ecd* classes for that year number back in **/home/<user>/adas/adf11**. It will overwrite *zcd*, *ycd* and *ecd* data already there. Both resolved and unresolved classes are handled and a running commentary is provided on progress and problems encountered. Two arguments may be given in the call to **generate_adf11_classes_10-12.pl** to alter the input subdirectory and the output subdirectory respectively.

4.3 Generating the superstage

condensation The creation of a data for a new child partition is largely automatic. It is in the province of ADAS416 and makes use of a script driver dataset located in **/home/<user>/adas/scripts416/**. The script driver design is standard and illustrated in figure 4.4. The main calculations are done by a FORTRAN code **adas416.for** in Offline-ADAS

```
tungsten

92
unresolved

parent pathways
---------------
gcr  :  /home/summers/adas/adf11/***92/***92_w_01.dat
pec  :  /home/summers/adas/adf15/pec89#w_01/pec89#w_01_pju#ne**.dat
gtn  :  /home/summers/adas/adf13/gtn89#w_01/gtn89#w_01_pju#ne**.dat
f-pec:  /home/summers/adas/adf40/fpec89#w_01/fpec89#w_01_pju#ne**.dat
f-gtn:  /home/summers/adas/adf42/fgtn89#w_01/fgtn89#w_01_pju#ne**.dat

partition specification
-----------------------
//#02/p00/ 00 01/
      p01/ 02 03 04 05/
      p02/ 06 07 08 09 10 11 12/
      p03/ 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27/
      p04/ 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45/
      p05/ 46 47 48 49 50 51 52 53 54 55/
      p06/ 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74/
//#01/p00/ 00/p01/ 01/p02/ 02/p03/ 03/p04/ 04/p05/ 05/p06/ 06/p07/ 07/
      p08/ 08/p09/ 09/p10/ 10/p11/ 11/p12/ 12/p13/ 13/p14/ 14/p15/ 15/
      p16/ 16/p17/ 17/p18/ 18/p19/ 19/p20/ 20/p21/ 21/p22/ 22/p23/ 23/
      p24/ 24/p25/ 25/p26/ 26/p27/ 27/p28/ 28/p29/ 29/p30/ 30/p31/ 31/
      p32/ 32/p33/ 33/p34/ 34/p35/ 35/p36/ 36/p37/ 37/p38/ 38/p39/ 39/
      p40/ 40/p41/ 41/p42/ 42/p43/ 43/p44/ 44/p45/ 45/p46/ 46/p47/ 47/
      p48/ 48/p49/ 49/p50/ 50/p51/ 51/p52/ 52/p53/ 53/p54/ 54/p55/ 55/
      p56/ 56/p57/ 57/p58/ 58/p59/ 59/p60/ 60/p61/ 61/p62/ 62/p63/ 63/
      p64/ 64/p65/ 65/p66/ 66/p67/ 67/p68/ 68/p69/ 69/p70/ 70/p71/ 71/
      p72/ 72/p73/ 73/p74/ 74/
---

child pathways
--------------
gcr  :  /home/summers/adas/adf11/***92/***92_w_02.dat
pec  :  /home/summers/adas/adf15/pec89_02#ne/pec89_02#ne_pju#ne**.dat
gtn  :  /home/summers/adas/adf13/gtn89_02#ne/gtn89_02#ne_pju#ne**.dat
f-pec:  /home/summers/adas/adf40/fpec89_02#ne/fpec89_02#ne_pju#ne**.dat
f-gtn:  /home/summers/adas/adf42/fgtn89_02#ne/fgtn89_02#ne_pju#ne**.dat

C-------------------------------------------------------------------------
C  sample scrip416 file
C
C  Notes:
C         (1) '***' in gcr pathway denotes 'acd', scd' etc
C         (2) '***' in pec, gtn, f-pec and f-gtn pathways denote 'llr', 'pjr'
C                   or pju.
C         (3) '**'  in pec, gtn, f-pec and f-gtn pathways denote partition
C                   member index.  This is the ion charge for the #01 root
C                   partition.  The indexing begins at 00
C         (4) Leave pathways which are not needed as blanks
C         (5) Choices at line 4 are 'resolved' or 'unresolved'
C         (6) Partitions are given in decreasing level order, that is
C             child before parent. This is the same rule as is used in
C             adf11 files with partition blocks and is required by the
C             partition block reader xxrptn.for.
C
C
C  Author:   Hugh Summers
C  Date  :   24 august 2005
C
C-------------------------------------------------------------------------
```

Figure 4.4: .

initiated from the IDL command line as

```
IDL>  z0 = 74
IDL>  class_list =        ['acd' 'scd' 'plt' ]
IDL>  adas416_script =    '$ADASHOME/adas/scripts416/partition_02_tungsten_1992.dat'
IDL>  run_adas416,        z0=z0, adas416_script=adas416_script
```

or

```
IDL>  run_adas416,        z0=z0, adas416_script=adas416_script, $
                          class_list=class_list
```

or

```
IDL>  run_adas416,        z0=z0, adas416_script=adas416_script, $
                          class_list=class_list,/adf11_only
```

where the second form restricts the *adf11* classes handled and the third restricts to *adf11* data only, omitting *adf15* and *adf40* processing. This overrides the script driver. Note that the procedure will terminate if the *adf11* classes *acd* and *scd* are not present in the unresolved case (or *acd*, *scd*, *qcd*, *xcd* in the resolved case - that is when the connection vector is non-trivial. The parameter *z0* is merely a check if the script driver does name the element explicitly.

Some technical details should be noted. ADAS data format specifications, especially *adf11*, *adf15* and *adf40* have been modified to cope with heavy elements and superstage condensation. Additional information on connection vectors and partition layers is present and some adjustment in field mnemonics and field use has been made. Updated access codes

xxdata_11.for
xxdata_15.for
xxdata_40.for

read the new datasets and all older versions transparently to the user. The reading routines fill in by inference and context the extra information for older data sets. Particularly, redundant or limited content fields in the transition header line of *adf15* files are used more fully. This is clearly desirable, since for example, a spectrum line associated with a superstage is well defined for computational spectral modelling and simulation by its numerical values and superstage but the spectral analyst may still wish to know what actual charge state that line is from. Also, the tail comment lines of ADAS data sets are written automatically and contain extensive information. The information has been associated with keywords. This means that the modern heavy element files have comment sections which can be read and analysed automatically. Extra subroutines, such as **xxrcmt_15.for** are now present in the ADAS libraries to do this. Automatic scanning for the preparation of tag files for OPEN-ADAS is also facilated by these changes. It is recommneded that ADAS users switch over to the new access routines as soon as possible. Details are in the appendices to this technical note.

Chapter 5

Lifting the baseline

The calculation of the atomic structure of an ion underpins all estimates of the radiation emitted from the element and its interaction with the plasma. Accurate structure data is thus an essential key ingredient of the prediction of emission from the plasma. The present work is concerned with establishing the infrastructure of atomic modelling for all heavy species. It must therefore be able to supply the appropriate derived data for any required species on a timescale matching the short period variations of fusion interest and need. This may be achieved by a carefully considered layered approach, trading coverage against precision. The picture pursued here is of moderate quality data with full coverage, restricted span data of higher precision targeting important diagnostic iso-electronic sequences and fiducial data for individual key ions. The infrastructure is required to support the seamless substitution of these improved approximations as and when they become available. In this work, these layers are called baseline, level 1 and level 2. Coordinated, collaborative work by others, outside the scope of this thesis, but under the umbrella of the ADAS Project, seeks to ensure that level 1 and level 2 data is correctly targeted and flows in appropriately. In chapters 4 and 5 of this work the targeting will be described in more detail. In this section, a brief overview of structure calculations is given, showing how they fit into this picture.

Ideally, energy levels, their quantum number labelling and transition probabilities would come from experimental measurements. For the present situation of heavy species, there is a paucity of such data and, even for a single targeted ion, they are generally incomplete (from the point of view of deducing total radiated line power). Also, such observational data, from a collisional perspective, only supplies information on the asymptotic behaviour of dipole allowed cross-sections. So the primary atomic structure inputs must come from theoretical calculations.

There are many atomic structure codes and particular preferences in various collaborating groups such as AUTOSTRUCTURE (University of Strathclyde, University College London and collaborators), HULLAC (Hebrew University Jerusalem, Lawrence Livermore National Laboratory), CIV3 (Queen's University Belfast), COWAN (Los Alamos National Laboratory, Oakridge National Laboratory, Auburn University, Strathclyde University and collaborators), GRASP (Oxford University, Queens University Belfast, Strathclyde University and collaborators), FAC (Harvard University and astrophysical collaborators), MCHF (Vanderbilt University), other MCDF (various). These various codes have characteristics which make them more or less suitable for the present purposes. As one progresses to higher ionization stages of a heavy species such as tungsten, the importance of the relativistic terms in the Hamiltonian increases with respect to the electrostatic and kinetic terms. Also, the angular momentum coupling schemes, appropriate for population and ionization state modelling, change in parallel.

In consequence, the COWAN code has been selected for the baseline production of the present work. Atomic structure codes can be readily extended with a free electron wave function calculation capability. This means that such codes can supply resonance capture/Auger data (within the independent resonance approximation) and the free-bound matrix elements required for radiative recombination without the complexity of a full collisional code. AUTOSTRUCTURE has been specially developed for these purposes, with dedicated post-processors for radiative and idelectronic recombination coefficients. It has been selected for production of data at level 1 and level 2 for the ADAS Project. Additionally AUTOSTRUCTURE sets up targets for level 1 and level 2 collision calculations (RMATRIX see section 2.2.2) and GRASP sets up targets for level 2 relativistic collision calculations (DARC see section 2.2.2) for the ADAS Project.

Code	Method	Usual application	Precision (E%, A%)	Comments
AUTOSTRUCTURE	Multi-config, Breit-Pauli, Thomas-Fermi and Slater-type parametric potential	General + Auger rates + Born integrals	(~2, ~5) typically dependent on CI scope.	Recently extended to multiply- occupied f-shells. Extended experience of use up to M-shell. Limited coupling scheme information. Specially tuned for dielectronic and radiative recombination. Can separate term and level resolution calculations. A preferred code for ADAS.
COWAN	Multi-config, Breit-Pauli, Hartree-Fock potential.	General + Auger rates + Born	(~2, ~5) typically dependent on CI scope and tuning.	Handles multiply-occupied f-states. Extended experience in many complex systems. Flexible coupling scheme information. Easy access to configuration average information. Executes level resolution calculation and averages to terms. A preferred code for ADAS.
HULLAC	Multi-configuration, Dirac Hamiltonian; j-j coupled basis, Breit and QED	General, but extensive use with EBIT measurements.	(~2, ~5) typically dependent on CI scope.	Proprietary code package; structure code part matched to distorted wave collision code and collisional-radiative modelling.
FAC	As for HULLAC	General, but mostly astro-physics.	(~2, ~5) typically dependent on CI scope and tuning.	Public domain variant of HULLAC. Use increasing and experience building up.
GRASP	Multi-configuration, Dirac/ Breit Hamiltonian; MCDHF or parametric potential; various couplings and optimizations.	General.	(<1, <3) with extensive core/ valance CI.	High grade code, but MCDHF not always able to converge on potential. Tuned to DARC fully relativistic version of R-matrix collision code. A preferred code for ADAS level 2 in relativistic region.

Figure 5.1: .

Electron impact excitation and ionization are summarized together, reflecting both physical situations in plasmas and computational developments. For ions of heavy species, with closed quantum shells in the valence n-complex and/or the adjacent lower n-shell, excitation to auto-ionising resonances is a major (and usually dominant) pathway to ionization. The direct ionization (inner and outer shell) is therefore often a smaller part of the total. Typically, such ions are dynamic influx ions, displaced to a thermal plasma regime above their natural stationary equilibrium location. Thus inner-shell excitation/ionization is futher favoured. From a theoretical point-of-view, precision calculations of excitation cross-sections take account of initially bound-electron 'flux losses' to the continuum. Modern close-coupling calculations seek to address this with pseudo-states and complete pseudo-state expansions. The latter allows deduction of the ionization cross-section and so the usual calculational separation into excitation and ionization is not appropriate.

It is convenient to introduce some notation for excitation and ionization at this point. Although excitation data is often presented as cross-sections, the more convenient quantity (and more immediate from the reactance matrix) is the dimensionless collision strength,Ω_{ij} , symmetric between initial state i and final state j and the Maxwell averaged collision strength $\Upsilon_{ij}(T_e)$. With projectile electron of energy ϵ_i and ϵ_j with the target in initial (lower energy) and final (upper energy) states respectively, and ΔE_{ij} the transition energy, then $\epsilon_i = \epsilon_j + \Delta E_{ij}$. Tabulations of collision strength are usually in terms of 'threshold parameter' X with the threshold. Then Υ_{ij} and has similar symmetry properties and the same threshold value as Ω_{ij}. The familiar excitation cross-section, de-excitation cross-section , excitation rate coefficient and de-excitation rate coefficient are then given by

It is noted that the collision strength from accurate theoretical calculation generally shows an elaborate resonance structure superposed on a smoother background. Typically many thousands of energy points are required to delimit fully the resonances. The Maxwell-averaging smoothes over the resonances to gave a reasonably slowly varying interpolable and moderate tabulation density suited to application in plasmas. Tabulations of collision strengths in principle allow calculation of rates for non-Maxwellian distributions. Unfortunately, collision strength tabulations in the general literature often have undefined implicit averaging over resonances. This is a flawed situation, which is avoided by tuning interval averaged collision strengths to the energy scale lengths of the distribution functions - a practice possible only with the fully delimited resonance structure of detailed cross-section calculations. It is noted that all electron collision strengths for fusion plasma application are total collision strengths, averaged over collision directions. The electron - ion collision processes in fusion plasma can be treated reliably as isotropic.

Excitation-autoionisation rates are simply excitation rates multiplied by an Auger yield branching factor. This factor can come from the atomic structure calculations, enabled for Auger rates described in the previous section. Direct ionization rates are integrated over the energy (possibly a weighted sum over continuum pseudostates) of the ejected electron and averaged over the energy of the incident electron. It is convenient for tabulation of ionization excitation rates to use a reduced rate coefficient which excludes the exponential factor and multiplies by the initial ion statistical weight. This makes a slowly varying interpolable quantity similar in character to the.

Code	Method	Usual application	Precision (%)	Comments
AUTOSTRUCTURE/ COWAN	Born with modified threshold region.	Low - medium/ high z.	(<40%)	Very general, stable and enabled by all structure codes with a free electron wave-function generator. No spin change. LS and IC coupling. Suitable for ADAS baseline.
CCC / CCC-R	Convergent close-coupling.	Low - medium/ high z; 1-2 valence electrons	(<5%)	Highest precision, inefficient for very many energies and delimiting resonances. Limited ion scope. Currently being extended to Dirac relativistic.
DARC/ DRMPS	Relativistic R-matrix close-coupling / with pseudo-states.	Low - high z.	(~ 5-10%)	Very high precision, tuned to GRASP structure and shared algebra. Resonances included. Recent pseudo-state extension increases heavy element near neutral scope. Use also for ionization. Intermediate coupling. Suitable for ADAS level 2 at low and high z.
HULLAC / FAC	Distorted wave.	Medium - high z.	(~20%)	Intermediate coupling, includes spin change, no resonances. Efficient algebra – but now used universally. Matched to HULLAC structure part.
RM / RMPS	R-matrix close-coupling / with pseudo-states.	Low - medium z.	(~ 5-10%)	High precision, tuned to AUTOSTRUCTURE and shared algebra. Resonances included. Use also for ionization. Implemented for isoelectronic sequences with scripts. LS coupling. Parallelized versions. Suitable for ADAS level 1, 2 medium-scale mass production.
RM – ICFT /RM-II	R-matrix close-coupling with internediate-coupling frame transformations/ R-matrix close-coupling with IC inner region.	Medium - medium/ high z.	(~ 5-10%)	As for RM, but extends to higher z ions in intermediate coupling. Suitable for ADAS level 1, 2 medium-scale mass production. R-matrix inner region IC gives improved higher z treatment. Suitable for ADAS level 2 and benchmarking of RM-ICFT.
RM-RD / DARC-RD	R-matrix close-coupling with radiation damping.	Medium - high z.	(~ 5-10%)	As for RM, but extends to high z ions with significant radiative/ Auger branching of resonances. Suitable for ADAS level 1, 2.
TDCC	Time-dependent close coupling.	Low z; 1-2 valence electrons.	(<5%)	Highest precision. Benchmark for low-z ionization. Used for ADAS level 2.
UCL-DW / JAJOM	LS distorted wave with IC transformation.	Medium -medium/ high z.	(~20%)	Matched to AUTOSTRUCTURE. Extension to IC via algebraic transformation. Includes spin change. No resonances. Can isolate calculation of cross-sections starting with selected metastables. Now inefficient and falling out of use in comparison with RM.

* depends on precision of multi-configuration multi-electron structure calculation and/or close-coupled set and/or pseudo-state span and completeness.

Figure 5.2: .

Turning to the actual calculations of electron impact cross-section and rate coefficient data, consider firstly ionization. In so far as ionization resembles a binary-encounter classical collision, there has been extensive use of variants of the Thomson Classical formula to provide a universal capability. The variants are typically based on a sum of Thomson-like shell contributions with adjustable parameters. These methods are surprisingly successful, especially those which adjust for classical exchange, non-classical (logarithmic) high energy behaviour and have a considered and flexible approach to effective shell ionization potentials and equivalent electrons. On top of such parametric form, global scaling is used to adjust results to match available higher quality data. ADAS baseline uses such methods, but is currently moving towards an improved baseline as large scale configuration-average distorted wave results are now becoming sufficiently comprehensive. This is discussed further in the more detailed ionization/recombination of the baseline description in a later chapter. In this brief review, it is the methods which can advance the description to level 1 and level 2 by substitution or as scaling fiducials which are of concern and to which we seek to draw attention. It is noted that that there are now four high precision method which can provide such fiducials, namely CCC (Flinders University), RMPS (Drake University, Belfast University), DRMPS (Strathclyde University) and TDCC (Auburn University). These are identified in Table 2 below. All achieve this within a close-coupling formulation, the first three through the completeness of a discrete orthogonal continuum-spanning pseudo-state basis, within an excitation code formulation. Participants and collaborators of the ADAS Project are developers of some of these methods and are selectively building the level 1 and level 2 ADAS base. Current computational power allows only one or two valence electron systems. But, as will be shown later, this is well suited to the key ionizations states for diagnostics distinguished in a superstage approach.

For excitation, Born, distorted wave and R-matrix methods are available. As is well known, excitation codes are linked to atomic structure codes, which provide the target ion description, and precision of the target is a prerequisite for precision of the cross-sections. Thus, the plane-wave Born approximation is of quite acceptable accuracy for the high-coverage baseline of the present work and very satisfactory for extended configuration average top-up when linked to codes such as COWAN and AUTOSTRUCTURE. As described earlier, the Born approach works better as we move to heavier species and higher charge states. This is because of the ionization balance shift and because the Born no spin change character is ameliorated by intermediate coupling spin system breakdown. It is also straightforward for all structure codes to return the Bessel integrals required for a Born approximation. Then more sophisticated calculations can be substituted for individual diagnostic ions and diagnsotic iso-electronic sequences. Table 2 summarizes some of the methods and their relevance for lifting the ADAS baseline of the present work to level 1 and level 2. The distorted wave methods (HULLAC, UCL-DW) improve on BORN, through their partial wave resolution and the associated ability to force unitarity, and by enabling spin changing transitions. Nonetheless, the inability to deal with resonances remains a problem and does not, for the ADAS Project purposes, lift the method sufficiently above Born. So ADAS looks to close-coupling methods for its higher precision cross-sections. The R-matrix method stands out because of its handling of resonances and its great efficiency in obtaining results at the very many energies required to delimit them. For heavy species and their range of ions, three effects matter. Moving to higher charge state, the relativistic terms in the Hamiltonian become more important and there is a shift to intermediate coupling. This effect is significant for precise fine structure wavelengths before it significantly changes fine-structure components cross-sections from their statistical proportions, the latter coming into play by z 18. Then two variants (RM-II and RM-ICFT) become relevant, the former being the complete intermediate-coupling model (within the Breit-Pauli formulation) and the latter an efficient approximation. The fully relativistic, that is Dirac/Breit approach is necessary for z¿50 provided by the DARC variant. For near-neutral charge states, the so-called flux loss to the continuum matters and the pseudostate methods RMPS, DRMPS and CCC come into play. DRMPS supports collisions with low charge states of very heavy species where the core is relativistic.

5.1 Global extensions to baseline data

The COWAN atomic structure code has proved very satisfactory for the ADAS heavy element baseline and has allowed easy engagement with many other specialised researchers on atomic structure and transition probabilities who use COWAN as a support vehicle. COWAN is however in an essentially frozen non-developmental state and has only the limited plane-wave-Born (PWB) extension into the collisional domain. In other aspects of ADAS, especially dielectronic recombination, ADAS uses the AUTOSTRUCTURE code. AUTOSTRUCTURE remains a developing code and has much stronger connection with the collisional domain. AUTOSTRUCTURE and CIV3 in Europe and the USA are probably the most preferred routes into R-matrix cross-section calculations. R-matrix alone can ultimately provide the collision cross-section precision to which ADAS aspires. In this section, exploitation of AUTOSTRUC-

TURE for global improvement of atomic structure and cross-section data is the main objective. Progess in specific targetting of key heavy elemnt ions is deferred to the subsequent section. In addressing theoretical atomic structure, it is universally recognised that, ab initio, spectroscopic precision of predicted transition wavelengths is not obtainable. It is this issue which inhibits the direct use of ADAS theoretical data in spectral wavelength analysis. The complexity, specificity and time consuming character of theoretical to experimental wavelength matching has not as yet been handled by a global expert system. In this section, consideration is given to some small steps in this direction. Although these considerations are global, that is usable for any element, to maintain the relevance of the central ADAS database to fusion and astrophysical applications, actual data tabulations in ADAS data formats are restricted to subsets of elements, which are called the light, medium, and heavy element sets as follow:

Light element set: Hydrogen - Zinc, inclusive.
Medium element set: Krypton, Molybdenum, Silver, Tin, Xenon, Cesium, Barium, Lanthanum, Neodymium, Gadolinium, Ytterbium.
Heavy element: Hafnium, Tantalum, Tungsten, Rhenium, Platinum, Gold, Lead, Radon.

These sets may be modified and/or extended according to changing relevance and need.

5.1.1 Ionisation potentials

For ionisation potentials, ADAS recognises the tabulations of the National Institute of Standards and Technology (NIST) Atomic Spectra Database [1] as its primary source for spectroscopic precision energy level data. Unadjusted theoretical atomic structure codes cannot match the precision of assessed NIST data. ADAS ionisation potentials are archived in ADAS data format *adf00*. Historically these datasets held simply the ionisation potentials of each ion, that is from the ground level (not term) of one ion to the ground level of the next higher ion, along with the full configuration specification of the ground level as a simple list against ion charge. It is convenient for other purposes now to add the outer quantum numbers (multiplicity, total L and total J). Thus for example in the tungsten data set *w.dat* the line of entry for W^{+11} in the data set appears as:

11 2.31602473d+02 1s2 2s2 2p6 3s2 3p6 3d10 4s2 4p6 4d10 4f13 5s2 5p2 (4)3(3.5)

comprising ion charge, ionisation potential, full configuration and outer qauntum numbers. Concerning previous versions of *adf00* datasets, the default data in *adf00* were taken from Carlson *et al* (1970) [**?**]. These data for light elements from hydrogen to neon and for selected other species have included for some years revisions from NIST. In particular the neutral and singly ionised stages of most elements were revised using NIST data in 2006. Then, NIST where available, was incorporated for the ions of elements aluminium, argon, calcium, chlorine, cobalt, chromium, copper, gallium, germanium, krypton, manganese, sodium, nickel, phosporus, scandium, sulphur, selenium, titanium, vanadium and zinc in 2011, from the work of Foster. It has been convenient for the present work to extend the NIST substitutions for all elements up to radon. In cases where NIST data on the ionisation potential are missing, the Carlson values are used. Where NIST does not specify the quantum numbers of the ground level, default (high Z) values are taken from *adf00_extension_ground_level_list.dat* which resides in the *adf00* directory. The comment section at the end of an element *adf00* dataset records the gaps and fill-in in a truth table and so is more informative than the previous updates (see appendix A for more details).

This updating of *adf00* will now be repeated periodically, perhaps every few years, as further NIST spectral assesments are completed. ADAS updating by hand is not feasible for the complete set of elements up to radon, so the procedure has been automated. The procedures are not appropriate for interactive ADAS, nor are they likely to be of interest to the general ADAS user. The procedures fit suitably into 'OFFLINE-ADAS' and are stored there. Access will normally be permitted only for collaborators in central ADAS update, but for completeness, they are described here. The ADAS/ NIST ionisation potential upgrade capability is implemented as PERL scripts in

/home/adas/offline_adas/adas1#1/scripts

Two scripts provide the required functionality. The first of these is concerned with acquisition of NIST data from the internet. Besides the immediate need for ionisation potentials, in the next section NIST data will again be required for more substantial energy level upgrading of *adf04* datasets. NIST energy level data is subject to frequent revision, so it is appropriate for ADAS to take a snapshot of required NIST data which is archived locally. This snapshot provides the NIST source up until completion of the next ADAS release, whereupon a fresh snapshot can be taken. The snapshot is held by element and coupling, that is level (*ic*) or term (*ls*), in non-public ADAS space as

[1] http://physics.nist.gov/PhysRefData/ASD/levels_form.html

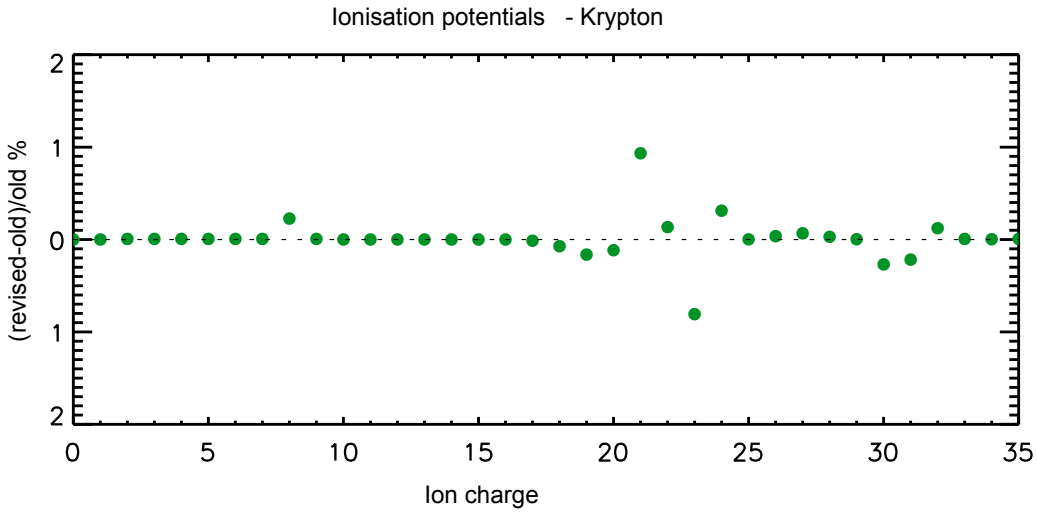

Figure 5.3: Comparison of revised and old krypton ionisation potential data. For ionisation stages Kr^{0} to Kr^{+9}, the old data already included NIST revisions. For ionisation stages Kr^{+10} to Kr^{+16}, there are no NIST data.For ionisation stages Kr^{+17} to Kr^{+35}, the old data already included NIST revisions. There are some modest changes in the NIST data since these earlier revisions.

/home/adas/nist_energy_level_tables/<element>/<coupling>#<ion>.dat.

The page for each ion is a cleaned simple ascii facsimile of the original NIST data. Subsequent action on this archive extracts the ionisation potentials and ground level quantum numbers for the base *adf00* element datasets. The scripts implementing these steps are

nist_get_data.pl	procures the NIST energy level data for an element from the internet, organises it for ADAS and archives it. A control parameter *coupling* taking values *ls* or *ic* determines if the extracted data is of term list or level list type (see section 5.1.2 for further discussion).The associated dataset names are *ls#<ion>.dat* and *ic#<ion>.dat* with with a arbitrary postfix such as *ls#<ion>_hps.dat* if required for distinguishing. Usually both *ls* and *ic* forms are produced at the same time. It is the level list form which is required for base *adf00* form production.
process_nist_to_adf00.pl	processes required NIST *ic* data for the ions of an element from the ADAS archive, obtains necessary fill-in information, prepares a complete *adf00* dataset in the revised format and files it in the ADAS database.

Typical calls, carried out by an updating person <uid> are

nist_get_data.pl	*−−nistroot=/home/adas/nist_energy_level_tables/ −−userpostfix=<uid> −−coupling=ic −−element=tungsten*
nist_get_data.pl	*−−nistroot=/home/adas/nist_energy_level_tables/ −−userpostfix=<uid> −−coupling=ls −−element=tungsten*
process_nist_to_adf00.pl	*−−nistroot=/home/adas/nist_energy_level_tables/ −−userpostfix=<uid> −−element=tungsten*

The revision, after verification, would be copied into central ADAS for general use and issued at a subsequent ADAS release. The PERL scripts require some tuning for a specific user's local Unix environment, firewalls etc. (see appendix D).

5.1.2 Atomic structure and energy levels

Although, as discussed earlier, ADAS recognises NIST as its primary source for spectroscopic precision energy level data, there are some problems. NIST energy level data are not complete and therefore cannot be mapped simply, one-to-one, as corrections, onto those from purely theoretical sources. Both NIST and the theoretical codes do not in

Figure 5.4: Analyis and comparison stages in adf04 dataset merging.

general give complete quantum number specifications of their levels. Also, configuration interaction and breakdown of coupling schemes mean that most of the quantum numbers used are not exact. Cross-matching of energy levels between NIST and theoretical codes and indeed between various theoretical calculations themselves, especially in the complex systems of the present work, is fraught. Recently the capabilities of the AUTOSTRUCTURE code have been extended to include production of plane-wave Born (PWB) and distorted wave (DW) collision cross-sections, and to work with kappa-averaged orbitals in the semi-relativisitic regime. These items, coupled with AUTOSTRUCTURE's close linkage/interfacing to ADAS and the ADAS data formats make a powerful addition to the tools for lifting the heavy element baseline. The collisional aspects will be discussed in the next sub-section. Here the point of note is that AUTOSTRUCTURE is the key code whose energy level outputs are to be aligned with NIST data and with other codes, especially COWAN. A programme of work has been initiated to that end. ADAS uses the data format *adf04* for energy level and transition data. As a first step towards energy level cross-matching between NIST, AUTOSTRUCTURE and COWAN, it is helpful to obtain the NIST energies for an ion in *adf04* format. A further script provides this functionality:

process_nist_to_adf04.pl	processes the required NIST data for the ions of an element from the ADAS/NIST archive, prepares an *adf04* dataset for each ion and files it in the ADAS database. The control parameter *coupling* is again present and the generated *adf04* data sets are of the form *ls#<ion>.dat* and *ic#<ion>.dat*. The processing is done by element. Note that the NIST *adf04* datsets contain no collisional rate coefficient datalines and so cannot support a population calculation.

Typical calls, carried out by an updating person <uid> are

process_nist_to_adf04.pl	*−−nistroot=/home/hps/nist_energy_level_tables/ −−coupling=ic −−userpostfix=hps –element=chlorine*
process_nist_to_adf04.pl	*−−nistroot=/home/hps/nist_energy_level_tables/ −−coupling=ls −−userpostfix=hps –element=chlorine*

By default, the data are stored in the directory */home/<uid>/adas_dev/adas/adf04/nist#<nuclear charge>/*. A number of difficulties may arise due to incompleteness of the original NIST data or because of the coupling schemes used for quantum number allocation in NIST. The latter problem is typified by the Ne-like ions, for which NIST may use *(j,j)* coupling assignments. ADAS *adf04* datasets, presently available in the database, and their access codes such as *xxdata_04.for* in FORTRAN, assume *LSJ* assignments. The *process_nist_to_adf04.pl* script in *ic* operation on encountering a *(j,j)* coupled level assigns blanks for the *2S+1* and *L* in the *adf04* dataset. In the *ls* case, there are only terms and no *J*, so a (j,j) coupled term would result in no entry for that term in the *ls* adf04 dataset. A remedial action is required which can be simply a 'by hand' editing of the *adf04* datasets, but more global semi-automatic action takes us to a new sophisticated *merge_nist_adas_adf04.pl* script and the Fortran codes associated with it. The process of merging NIST energy level data from a NIST *adf04 ic* dataset with a comprehensive theoretical *adf04* calculation for the same ion, if successful, in principle identifies the *LS* quantal assignments.

The scripted package *merge_nist_adas_adf04.pl* has been set up to analyse, compare and attempt to merge *adf04* datasets. This package serves a number of purposes in ADAS. Here, its use is only directed at the energy level lists of *adf04* datasets and not at the comparison and merging of transition lines. Pairwise comparisons of NIST/ AUTOSTRUCTURE, NIST/ COWAN and AUTOSTRUCTURE/ COWAN are of interest. As indicated in the schematic of figure 5.4, comparison is done progressively for parents, orbitals, configurations and levels. The first two, which are linked to structure code constructs, are not fully supported by NIST adf04 datasets. It is the configuration and level matching steps which are of principal concern and both are more difficult and subject to more substantial doubt in the NIST/ AUTOSTRUCTURE and NIST/ COWAN conparisons than in the AUTOSTRUCTURE/ COWAN case. It is necessary to take advantage of the completeness of the theoretical calculations and their symmetry ordered level energies and configuration averaged energy centroids to identify gaps in the NIST data and their approximate locations. Then no-crossing rule assumptions may be made within a $J - \pi$ symmetry for matching. These strategies are built into the key FORTRAN subroutines *g5cmch.for* and *g5lmch.for* using information from *g5alyz.for* (see figure 5.4). A downward progression from high to low *J* assists in the level assignments to terms and configurations. A last strategy is the ordering of terms of partially filled shells, which again can come from the theoretical structure calculations of parents, grandparents and so on. One of the outputs from the package is a text file summarising the analysis and matching. It is important that this output be reviewed before confidence is placed on the matching. Focussing now on

NIST in comparison with AUTOSTRUCTURE calculations, with successful matching, the ADAS strategy is for NIST energy level substitution to form a new composite *adf04* dataset which adopts the radiative and collisional transition data from the AUTOSTRUCTURE *adf04* dataset, but with energy level precision improvements from NIST. Details of the AUTOSTRUCTURE collisional rate advances from the ADAS baseline follows in the next sub-section. The concern here is the energy level structure. A typical call, carried out by an updating person <uid> is

process_merge_nist_adf04.pl −−nistroot=/home/hps/adas_dev −−adasroot=/home<uid>/adas_dev −−source=autos −−collision_type=pwb −−coupling=ic −−userdircode=cophps −−ion=cl7

This example of the neon-like ion Cl^{+7} is informative and may be usefully compared with the neighbouring iso-electronic sequence member S^{+6}. The top of the information text file, placed in the user's /adas/pass directory in and called *merge_nist_adf04_cl7.txt*, commences with a summary of the names and the presence or otherwise of the various sectional contents of the two *adf04* datasets. Then the parents, from the top line of the *adf04* datsets are compared.

```
file1:  /home/hps/adas_dev/adas/adf04/nist#17/ic#cl7.dat
file2:  /home/hps/adas_dev/adas/adf04/cophps#ne/pwb/ic#cl7.dat

llpref:  T
ltpref:  F

                    file1            file2
                    -----            -----
adf type:             3                3
no.of parents:        1                1
no.of levels:        16               89
no.of temps:          0               14
no.of trans:          0             3785

parents set:          T                T
frac prntge set:      F                F
orbitals set:         F                T
e-trans present:      F                T
p-trans present:      F                F
r-trans present:      F                F
h-trans present:      F                F
i-trans present:      F                F
s-trans present:      F                F
l-trans present:      F                F

 File 1 - file 2 parent cross-matching
----------------------------------------
  Master         Parent    File1   Energy (cm-1)   File2   Energy
  index          config.   index                   index   Deficit
  ------        -------   ------  --------------   ------  ---------
    1            (1S)        1      2809100.0        1      -180.0
```

The next section is a comparison of the configurations present. Note the configurations are given in Eissner notation for compactness. The NIST data set in fact has very few levels, such that in no case are all the expected levels of a configuration present. On the other hand the theoretical dataset is complete in all configurations. Thus, although there should be an energy deficit from the NIST dataset for every configuration, apart from the ground level, the script can not assess a mean shift of any of the configurations between the first and and the second data sets. So the shift is set to zero and a warning is given that net configuration shifts for matching cannot be done. Only an average level shift is possible from matched levels and this is noted in a further warning. Note that the '#' symbol indicates where one dataset includes a configuration but the other does not.

```
File 1 - file 2 configuration cross-matching
------------------------------------------------
 Master      Eissner              Energy (cm-1)   Level   File1   File2     Energy    level ct.
 index       config.                              count   index   index     deficit    deficit
 ------      -------------------  -------------   -----   -----   -----     -------   --------
   1        521522563                       0.0       1       1       1         0.0          0
   2        521522553514              1696905.0       2       2       2         0.0         -2
   3        521522553516              1996720.0       3       3       4         0.0         -9
   4        521522553517              2248100.0       2       4       5         0.0         -2
   5        521522553519              2362685.0       2       5       8         0.0        -10
   6        521512563515              2386675.0       2       6      10         0.0         -2
   7        52152255351D              2541556.4       3       7       #
   8        52152255351E              2553249.0       1       8       #
   9        521522553515              1835442.8      10       #       3         0.0        -10
  10        521512563514              2272597.3       2       #       6         0.0         -2
  11        521522553518              2311936.6      10       #       7         0.0        -10
  12        52152255351A              2391600.4      12       #       9         0.0        -12
  13        521512563516              2570045.0       4       #      11         0.0         -4
  14        521512563517              2825861.7       2       #      12         0.0         -2
  15        521512563518              2872502.2       4       #      13         0.0         -4
  16        521512563519              2933019.1       4       #      14         0.0         -4
  17        52151256351A              2952099.2       4       #      15         0.0         -4

Level count deficit between dataset 1 and dataset2 from configuration list =-21

###### No configuration shift applied to levels ######

###### Mean level shift =   -14527.7 applied to non-NIST levels ######
```

Contrast this with the corresponding table for Ar^{+8} shown below, where there are matching complete configurations between NIST and the theoretical dataset. From these matched configurations, a mean shift is evaluated which is applied to those PWB configurations not present in NIST and to those configurations where NIST is incomplete.

```
File 1 - file 2 configuration cross-matching
------------------------------------------------
 Master      Eissner              Energy (cm-1)   Level   File1   File2     Energy    level ct.
 index       config.                              count   index   index     deficit    deficit
 ------      -------------------  -------------   -----   -----   -----     -------   --------
   1        521522563                       0.0       1       1       1         0.0          0
   2        521522553514              2036018.7       4       2       2    -15317.0          0
   3        521522553515              2181173.2      10       3       3    -15845.5          0
   4        521522553516              2371317.8      12       4       4    -18736.3          0
   5        521512563514              2627280.3       2       5       5    -45054.3          0
   6        521522553517              2707355.8       3       6       6    -22812.7         -1
   7        521522553518              2763770.5       6       7       7    -22812.7         -4
   8        521512563515              2788737.7       3       8       8    -22812.7         -1
   9        521522553519              2833622.9      12       9       9    -19110.4          0
  10        52152255351A              2854523.6       9      10      10    -22812.7         -3
  11        52152255351B              2981700.3       4      11       #
  12        521512563516              2992998.0       1      12      11    -22812.7         -3
  13        52152255351C              3018343.7       3      13       #
  14        52152255351D              3042472.1      10      14       #
  15        52152255351E              3057371.0       9      15       #
  16        521512563517              3340440.9       2       #      12    -22812.7         -2
  17        521512563518              3393617.8       4       #      13    -22812.7         -4
  18        521512563519              3462048.9       4       #      14    -22812.7         -4
  19        52151256351A              3485341.8       4       #      15    -22812.7         -4

Level count deficit between dataset 1 and dataset2 from configuration list = 14

###### Mean configuration shift applied to levels ######
```

Finally the level match is made as shown below. The *2S+1* and *L* quantum numbers for the NIST levels (up to 12) come from the AUTOSTRUCTURE dataset. Levels 13 and 14 in NIST are *(j,j)* coupled but are from a comfiguration not included in the AUTOSTRUCTURE case. Levels 15 and 16 in NIST are given as *ic* coupled but again the configuration is not included in the AUTOSTRUCTURE case. Remaining levels are in the AUTOSTRUCTURE dataset, but not in NIST. Their energy levels have the mean level shift applied.

```
 File 1 - file 2 level cross-matching
-----------------------------------------
  Master          Level          Eissner            File1   Energy (cm-1)   File2   Energy
  index     2S+1    L       J    config.            index                   index   Deficit
  ------    ----------------   ----------------   ------  -------------   ------  ---------
    1         1     0      0.0   521522563            1            0.0       1         0.0
    2         3     1      1.0   521522553514         2      1689450.0       3    -14394.3
    3         1     1      1.0   521522553514         3      1704360.0       5    -15315.7
    4         3     1      1.0   521522553516         4      1972390.0      17    -16801.2
    5         3     2      1.0   521522553516         5      1997040.0      23    -18202.2
    6         1     1      1.0   521522553516         6      2020730.0      27    -24030.2
    7         3     1      1.0   521522553517         7      2242000.0      29    -17254.7
    8         1     1      1.0   521522553517         8      2254200.0      32    -17327.5
    9         3     1      1.0   521522553519         9      2356820.0      45     -6723.1
   10         3     2      1.0   521522553519        10      2368550.0      51     -6556.4
   11         3     1      1.0   521512563515        11      2371580.0      65    -27740.5
   12         1     1      1.0   521512563515        12      2401770.0      71     -9986.2
   13                      1.0   52152255351D        13      2521750.0       #         0.0
   14                      1.0   52152255351D        14      2534080.0       #         0.0
   15         1     4      4.0   52152255351E        15      2553249.0       #         0.0
   16         1     3      3.0   52152255351D        16      2553249.0       #         0.0
   17         3     1      2.0   521522553514         #      1683554.8       2
   18         3     1      0.0   521522553514         #      1696589.2       4
   19         3     0      1.0   521522553515         #      1790799.7       6
   20         3     2      3.0   521522553515         #      1810966.2       7
   21         3     2      2.0   521522553515         #      1812386.4       8
   22         3     2      1.0   521522553515         #      1816952.7       9
   23         3     1      2.0   521522553515         #      1822509.6      10
   24         1     1      1.0   521522553515         #      1827019.3      11
   25         3     1      0.0   521522553515         #      1830135.6      12
   26         1     2      2.0   521522553515         #      1831326.7      13
   27         3     1      1.0   521522553515         #      1832541.4      14
   28         1     0      0.0   521522553515         #      1912993.4      15
   29         3     1      0.0   521522553516         #      1973012.5      16
    .         .     .
   91         3     3      3.0   52151256351A         #      2937499.2      87
   92         3     3      4.0   52151256351A         #      2937536.0      88
   93         1     3      3.0   52151256351A         #      2937756.5      89


Level count deficit between dataset 1 and dataset 2 from level list =   4


###### Config. level count deficit mismatch  with actual level count deficit ######
```

The script also produces the final merged *adf04* dataset, with ordered energy levels, following NIST when available or with the adjusted theoretical energies otherwise. The transition data comes from the theoretical dataset. Levels present in NIST but not present in the theroretical dataset are omitted. The dataset is written to the user's /adas/pass directory as *merge_nist_adf04_cl7.dat* for inspection and archiving if acceptable.

Returning to the issue of the *adf00* datasets of the section 5.1.1, there are two more complex variants of the base adf00 data class. These are designed to support generalised collisional radiative (*GCR*) studies. They contain ionisation potentials and quantum number assignments of both ground and metastable states of ions. They occur in both *ls* and *ic* variants with dataset names of the form *<elem. symb.>_ls.dat* and *<elem. symb.>_ls.dat*. With the NIST *adf04* archive available with *ic* quantum numbers given, two scripts can prepare these additional *adf00* datasets as follow:

make_nist_ic_adf00.pl
make_nist_ls_adf00.pl

Typical calls, carried out by an updating person <uid> are

make_nist_ic_adf00.pl *−−nistroot=/home/<uid>/adas_dev/adas/ −−userpostfix=<uid>*
−−element=chlorine
make_nist_ls_adf00.pl *−−nistroot=/home/<uid>/adas_dev/adas/ −−userpostfix=<uid>*
−−element=chlorine

The above scripts operate on NIST *adf04* datasets and cannot go correctly to completion if any one of these datasets does not include the required metastables and/or their quantum numbers. This may be the case especially for low charged members of the neon-like and argon-like sequences. Such NIST *adf04* datasets in central ADAS have been modified sufficiently to provide this completeness as far as argon, using the information from the 'merge' analysis above. Without these modifications, the script and *adf00* dataset production will occur, but the number of identified metastables, for example for the neon-like ionisation stage, could be incomplete. **It is important therefore to interpret the information correctly in *ls* and *ic* resolved *adf00* datasets.** Consider the unmodified *ls* dataset for argon below

```
argon             -18  2  2  4  3  4  2  2  1  2  2  4  3  4  2  2  1  2  1 /wf=   0.00001/
 0     1.58187697d+01  1s2  2s2  2p6  3s2  3p6         ( 1)0(  0.0)
    1  0.00000000d+00  1s2  2s2  2p6  3s2  3p6         ( 1)0(  0.0)
 1     2.76380961d+01  1s2  2s2  2p6  3s2  3p5         ( 2)1(  2.5)
    1  0.00000000d+00  1s2  2s2  2p6  3s2  3p5         ( 2)1(  2.5)
    2  1.63656692d+01  1s2  2s2  2p6  3s2  3p4  3d1    ( 4)2(  9.5)
 2     4.06674166d+01  1s2  2s2  2p6  3s2  3p4         ( 3)1(  4.0)
    1  0.00000000d+00  1s2  2s2  2p6  3s2  3p4         ( 3)1(  4.0)
    2  1.66942360d+00  1s2  2s2  2p6  3s2  3p4         ( 1)2(  2.0)
    3  4.05682479d+00  1s2  2s2  2p6  3s2  3p4         ( 1)0(  0.0)
 3     5.97531254d+01  1s2  2s2  2p6  3s2  3p3         ( 4)0(  1.5)
    1  0.00000000d+00  1s2  2s2  2p6  3s2  3p3         ( 4)0(  1.5)
    2  2.62450980d+00  1s2  2s2  2p6  3s2  3p3         ( 2)2(  4.5)
    3  4.33611158d+00  1s2  2s2  2p6  3s2  3p3         ( 2)1(  2.5)
 4     7.48555059d+01  1s2  2s2  2p6  3s2  3p2         ( 3)1(  4.0)
    1  0.00000000d+00  1s2  2s2  2p6  3s2  3p2         ( 3)1(  4.0)
    2  1.84949204d+00  1s2  2s2  2p6  3s2  3p2         ( 1)2(  2.0)
    3  4.52925416d+00  1s2  2s2  2p6  3s2  3p2         ( 1)0(  0.0)
    4  1.02566544d+01  1s2  2s2  2p6  3s1  3p3         ( 5)0(  2.0)

.    .    .

16     4.12066549d+03  1s2                           ( 1)0(  0.0)
    1  0.00000000d+00  1s2                           ( 1)0(  0.0)
    2  3.10414847d+03  1s1  2s1                      ( 3)0(  1.0)
17     4.42622267d+03  1s1                           ( 2)0(  0.5)
    1  0.00000000d+00  1s1                           ( 2)0(  0.5)
c-------------------------------------------------------------------------------------------
c
c  New updated adf00 ls format dataset with current NIST values.
c  Replaces earlier dataset of same name.
c
c  Sources:    /home/hps/adas_dev/adas/adf04/nist#18/    ls and ic forms
c              /home/hps/adas_dev/adas/adf00/            base form
c  Code:       make_nist_ls_adf00.pl
c  Producer:   hps
c  Date:       03-Sep-2012
c
c  Expected metastables are not satisfied by NIST adf04 datasets for ionisation stages:
c           Ion charge      Deficit
c                0            1
c                2            1
c
c-------------------------------------------------------------------------------------------
```

The top line (lines if there are more than 18 ionisation stages) vector following the (signed) nuclear charge gives the expected number of metastables for each ionisation stage. This is according to the ADAS *GCR* prescriptions as given by the IDL proecedure *metastable.pro* and the PERL function *metastable.pm*. The actual metastables present, that is found in the NIST *adf04* datasets, are indexed in the body of the dataset. In the comments section of the dataset, an advisory message gives the deficient ionisation stages, if any, and the deficit. A further point to note is the *wf*= entry on the top line of resolved *adf00* datasets. The zero energy reference point is the lowest level of the neutral ionisation stage of an atom. In the base *adf00* datasets, the ionisation energy of the neutral atom is with respect to this energy and is to the lowest level of the singly ionised atom. By contrast, in the *ls* resolved *adf00* dataset, the ionisation energy of the neutral is from the lowest term (not level) of the neutral to the lowest term of the singly ionised stage. Consistent energetics means that there is a 'work function' to deliver the neutral ground term atom to the plasma. This is the *wf*= entry and is non-zeo for *ls* type *adf00* datasets. Full description of the extended format metastable resolved *adf00* datasets are given in appendix A.1.

5.1.3 Collision cross-sections

The AUTOSTRUCTURE code of Badnell, one of the key fundamental codes underpinning ADAS, now has a capability for electron impact excitation cross-section calculation within the plane-wave-Born (*pwb*) and distorted wave (*dw*) approximations. This is an important development. AUTOSTRUCTURE has long been familiar to ADAS users as the source of the very extensive *adf09* database of state-resolved dielectronic recombination data. Less familar may be the matching radiative recombination coefficients (*adf48*), photo-excitation data (*adf38*) and photoionisation data (*adf39*). AUTOSTRUCTURE is also one of the preferred structure codes which front-end R-matrix calculations. With the new collisional extensions, AUTOSTRUCTURE covers the capabilities of the COWAN code for the ADAS *adf04* baseline (*pwb*), but takes ADAS forward into more refined collisional cross-section data in a manner fully consistent with much other data in ADAS. It is this exploitation of AUTOSTRUCTURE for ADAS as a global 'lift' of the database which is followed through here. As in the previous subsections 5.1.1 and 5.1.2, the machinery is operated through scripts

which, in this case, reside in the offline ADAS package */home/adas/offline_adas/adas7#3/*.

ADAS already has some precedents for mass data production using AUTOSTRUCTURE. The data format *adf27* is used to archive driver input datasets for AUTOSTRUCTURE runs with subdirectories */dr*, */rr*. */pe* and */pi* for dielectronic recombination, radiative recombination, photoexcitation and photoionisation. Nested sub-directories within these categories are firstly by iso-electronic sequence and then usually by producer/year number as, for example */adf27/dr/nelike/nrb00#ne* with final individual driver dataset named by ion, coupling (and possibly other parameters) such as *mo32ic23-3.dat*. For this new development,additional sub-directory categories */dw*, *pwb* and *dw_bbgp* are introduced, for the three methods (*dw*, *pwb* and *dw_bbgp*) of concern here, with a similar pattern of further nested sub-directories. Thus the dataset *~/adf27/dw/silike/cophps#si/ic#cu15.dat* is the driver for an AUTOSTRUCTURE distorted wave calculation for Cu^{+15} in intermediate coupling (*ic*) approximation. It is helpful for large scale production to execute similar calculations for all members of an isoelectronic sequence, so that continuity of behaviour with ion charge, *z*, can be verified and exploited interpolatively. It is convenient to store template drivers for each sequence in the same lowest level directory as the final drivers for the sequence. In the above silicon-like case, these are *template_ls* and *template_ic* as shown below. The templates are prepared by hand editing following the presciptions of the AUTOSTRUCTURE online guides. Note the fields delimited by <> brackets. These fields are substituted by actual parameters to obtain the fully specified driver dataset. In the case of the *dw* templates shown, the results of the AUTOSTRUCTURE *pwb* runs final *adf04* datasets are required in advance, from which certain of the parameter fields are extracted. Attention is drawn to the *NMETA* and *NMETAJ* parameters in the *ls* and *ic* templates respectively. These specify the highest term (level) up to which collisonal data are generated. This information is not known *a priori* before a structure run. An initial AUTOSTRUCTURE *pwb* run is made with these values set to 1 for a single ion. Inspection of the output *adf04* datasets shows the total number of terms (levels) which may be substituted in the templates. These values apply to all members of the isoelectronic sequence.

```
A.S. Si-like <ion> structure - energies + radiative rates + dw adf04 type 5
 &SALGEB RUN='DE' CUP='LSR' KCOR1=1 KCOR2=1 NMETA=325 MXCONF=18 MXVORB=9  &END
2 0  2 1  3 0  3 1  3 2  4 0  4 1  4 2  4 3
  2    6    2    2    0    0    0    0    0
  2    6    2    1    1    0    0    0    0
  2    6    2    1    0    1    0    0    0
  2    6    2    1    0    0    1    0    0
  2    6    2    1    0    0    0    1    0
  2    6    2    1    0    0    0    0    1
  2    6    1    3    0    0    0    0    0
  2    6    1    2    1    0    0    0    0
  2    6    1    2    0    1    0    0    0
  2    6    1    2    0    0    1    0    0
  2    6    1    2    0    0    0    1    0
  2    6    1    2    0    0    0    0    1
  2    6    1    1    2    0    0    0    0
  2    6    0    4    0    0    0    0    0
  2    6    0    3    1    0    0    0    0
  2    6    0    2    2    0    0    0    0
  2    5    2    3    0    0    0    0    0
  2    5    2    2    1    0    0    0    0
 &SMINIM NZION=<iz0> ORTHOG='NO' JPRINT=-33 MAXE=<maxe> &END
 &SRADCON MENG=-14 EMIN=<emin> EMAX=<emax> NDE=3 MENGI=-1  &END
<ndelist>
```

```
A.S. Si-like <ion> structure - energies + radiative rates + dw adf04 type 5
 &SALGEB RUN='DE' CUP='ICR' KCOR1=1 KCOR2=1 NMETAJ=725 MXCONF=18 MXVORB=9  &END
2 0  2 1  3 0  3 1  3 2  4 0  4 1  4 2  4 3
  2    6    2    2    0    0    0    0    0
  2    6    2    1    1    0    0    0    0
  2    6    2    1    0    1    0    0    0
  2    6    2    1    0    0    1    0    0
  2    6    2    1    0    0    0    1    0
  2    6    2    1    0    0    0    0    1
  2    6    1    3    0    0    0    0    0
  2    6    1    2    1    0    0    0    0
  2    6    1    2    0    1    0    0    0
  2    6    1    2    0    0    1    0    0
  2    6    1    2    0    0    0    1    0
  2    6    1    2    0    0    0    0    1
  2    6    1    1    2    0    0    0    0
  2    6    0    4    0    0    0    0    0
  2    6    0    3    1    0    0    0    0
  2    6    0    2    2    0    0    0    0
  2    5    2    3    0    0    0    0    0
  2    5    2    2    1    0    0    0    0
 &SMINIM NZION=<iz0> ORTHOG='NO' JPRINT=-33 MAXE=<maxe> &END
 &SRADCON MENG=-14 EMIN=<emin> EMAX=<emax> NDE=4 MENGI=-1  &END
<ndelist>
```

Output datasets for AUTOSTRUCTURE require further processing, for example to convert cross-sections to rate coefficients. From the ADAS point-of-view, the final deliverables are fully specified *adf04* datasets. In this case, the sub-directory structure is simpler than for *adf27*. The producer/year number sub-directory name appears first then below that lie the same sub-directory categories */dw*, */pwb* and */dw_bbgp* as for the *adf27* drivers as for example */home/adas/adas/adf04/cophps#al/pwb* for the aluminium-like sequence. The actual datasets then follow. The ADAS *adf04* dataset with which the typical user is familiar is properly called *adf04 type 3*. There are other types, namely 1, 3, 4, 5 and 6. which hold different kinds of collisional data. An AUTOSTRUCTURE run of *pwb*, *dw* or *dw_bbgp* method produces different *adf04* types. Thus *pwb* produces type 1 (containing collisions strengths as a function of final energy) and type 3 (the normal rate parameter type). *dw* produces type 5 (containing collision strengths as a function of initial energy) and type 3. *dw_bbgp* produces type 6 (partial wave collision strengths a function of *l*). Normal ADAS notation is to postfix the type to the dataset name for types 1, 4, 5 and 6 and to omit it for type 3. So final type 3 data sets such as *ic#si1.dat*, *ls#si1.dat*, *ic#si1_t1.dat* and *ls#si1_t1.dat* result for the aluminium like ion Si^{+1} in an AUTOSTRUCTURE *pwb* calculation.

Turning to the actual processing, relevant PERL scripts are in the */home/adas/offline_adas/adas7#3/scripts* sub-directory as follow:

setup_isoseq_pwb_adf27.pl	prepares fully specified *pwb* drivers from template.
setup_isoseq_dw_adf27.pl	prepares fully specified *dw* drivers from template.
setup_isoseq_dw_bbgp_adf27.pl	prepares fully specified *pdw_bbgp* drivers from template.
process_ion_pwb_adf27_to_adf04.pl	perform AUTOSTRUCTURE *pwb* run for an ion using its driver.
process_ion_dw_adf27_to_adf04.pl	perform AUTOSTRUCTURE *dw* run for an ion using its driver.
process_ion_dw_bbgp_adf27_to_adf04.pl	perform AUTOSTRUCTURE *dw_bbgp* run for an ion using its driver.
adas7#3_pwb_llbatch.pl	run AUTOSTRUCTURE *pwb* distributed batch processing for all the ions of an isoelectronic sequence.
adas7#3_dw_llbatch.pl	run AUTOSTRUCTURE *dw* distributed batch processing for all the ions of an isoelectronic sequence.
adas7#3_dw_bbgp_llbatch.pl	run AUTOSTRUCTURE *dw_bbgp* distributed batch processing for all the ions of an isoelectronic sequence.

A description of each script and how to use it is given in comment text at the head of the script. An AUTOSTRUCTURE *adf04* calculation for a multi-electron ion, dependent on the actual configurations included, can be quite large and time-consuming. For example, a phosphorus-like ion could take a week on a typical machine. It is appropriate therefore to distribute the processing for the members of an iso-electronic sequence over many machines. LOADLEVELLER provides this capability on the JAC Linux workstations at the EFDA-JET Facility. The last set of three scripts set up and launch these jobs in parallel. LOADLEVELLER notifies by email message when each ion of the sequence completes and routes the *adf04* output datasets to the ADAS database. A summary (*adas7#3_<method>_llbatch_<ion>.out*) and error information (*adas7#3_<method>_llbatch_<ion>.err*) is sent to the user's */pass* directory, . It is important to check that the *.err* file is empty. The AUTOSTRUCTURE code produces very extensive information (the so-called *.olg* file), including advice on re-dimensioning if insufficient space has been

allocated for the calculation, but in distributed batch processing, this is lost. A typical series of command line calls might then be

setup_isoseq_pwb_adf27.pl *−−adasroot=/home/hps/adas_dev −−userdircode=cophps −−isoseq=al −−element_category=light*

to set up the individual *adf27* drivers for *pwb*, followed by

process_ion_pwb_adf27_to_adf04.pl *−−adasroot=/home/hps/adas_dev −−userdircode=cophps −−isoseq=al −−ion=s3*

for the single aluminium-like ion S^{+3}, so obtaining the total level counts *NMETA* and *NMETAJ* from the output datasets

/home/hps/adas_dev/adas/adf04/cophps#al/pwb/ls#s3.dat and

/home/hps/adas_dev/adas/adf04/cophps#al/pwb/ic#s3.dat respectively.

These are edited into the aluminium-like *adf27 pwb*, *dw* and *dw_bbgp* templates. Then the commands

setup_isoseq_pwb_adf27.pl *−−adasroot=/home/hps/adas_dev −−userdircode=cophps −−isoseq=al −−element_category=light*

adas7#3_pwb_llbatch.pl *−−adasroot=/home/hps/adas_dev −−userdircode=cophps −−isoseq=al*

regenerate the fully specified *pwb* drivers and set the *pwb* batch processing going offline. Successful completion then allows the *dw* processing as

setup_isoseq_dw_adf27.pl *−−adasroot=/home/hps/adas_dev −−userdircode=cophps −−isoseq=al −−element_category=light*

adas7#3_dw_llbatch.pl *−−adasroot=/home/hps/adas_dev −−userdircode=cophps −−isoseq=al*

firstly to generate the drivers and then set the batch processing going offline. If required the *dw_bbgp* commands may then be executed in the same way. This is our normal mass production procedure. The scripts are tuned to working with LOADLEVELLER at EFDA-JET. Retuning of the scripts for batch processing on different systems is required.

5.2 Targetted extensions to heavy element baseline data

Bibliography

[1] A. Thoma, K. Asmussen, R. Dux, K. Krieger, A. Herrmann, B. Napiontek, R. Neu, J. Steinbrink, M. Weinlich, U. Wenzel and the ASDEX Upgrade Team. 'Spectroscopic measurements of tungsten erosion in the ASDEX Upgrade divertor'. Plasma Phys. Control. Fusion, **39**(9) (1997) 1487–1499. doi:10.1088/0741-3335/39/9/014

[2] J. Spence and H. P. Summers. 'The recombination and level populations of ions. III. The role of charge exchange from neutral hydrogen'. J. Phys. B, **19**(22) (1986) 3749–3776. doi:10.1088/0022-3700/19/22/018

[3] P. C. Souers. *Hydrogen Properties for Fusion Energy*. University of California Press (1986). ISBN 0-520-05500-4

[4] H. P. Summers (2007). Atomic Data and Analysis Structure User Manual. Available from: http://www.adas.ac.uk

[5] H. P. Summers, W. J. Dickson, M. G. O'Mullane, N. R. Badnell, A. D. Whiteford, D. H. Brooks, J. Lang, S. D. Loch and D. C. Griffin. 'Ionization state, excited populations and emission of impurities in dynamic finite density plasmas: I. The generalized collisional-radiative model for light elements'. Plasma Phys. Control. Fusion, **48**(2) (2006) 263–293. doi:10.1088/0741-3335/48/2/007

Appendix A

ADAS data formats

A.1 *adf00*: configurations and ionisation potentials

The basic data sets, of stage-to-stage form provide the ground configurations and ionisation potentials of every ion of every element up to lead. There is a second category of metastable resolved LS type which include metastable configurations and excitation energies between metastables as well as ionisation energies.

Data mnemonic:
Data root: **/home/adas/adas/adf00/**
Last update: Jan 18, 2007
Utilising subroutines: **xxdata_00.for**

Formatted files to adf00 specification:

Element	Members	Datasets	Quality
<elem. symbol>	H - Pb	<elem. Symbol>.dat	medium/high
<elem. symbol>	H - Ne, Ar	<elem. Symbol>_ls.dat	high

Notes: The format leaves open the opportunity for a intermediate coupling, J-resolved metastable form as <**elem. symb**>_**ic.dat**. GCR modelling in ADAS at this time only extends to LS resolved. The presently extended format is designed to support the needs of 'superstaging'. Spread sheet analyses of the metastable definitions and energy determinations are available and may be provided on request in the **/adf00** directory.

Data lines (stage to stage form):

Data lines	Format
ELEMENT, IZ0S	1a16,1i5
for I=0,|IZ0S|-1	
if (IZ0S.lt.0) then	
IZ, EION(I+1),CFG(I+1)	i2,1f16.8,1a120
else	
IZ,CFG(I+1)	i2,1a120
endif	
endfor	

Variable identification:

Name	Meaning	Comment
ELEMENT	element name	
IZ0S	signed nuclear charge	<0 => data lines include ionisation potential >0 => data lines do not include ionisation potential)
IZ	ion charge	
EION()	ionisation potential	eV
CFG()	configuration string	note precise formatting with 5 character space allocation to each shell

Sample 1: **/home/adas/adas/adf00/ne.dat**

```
.
neon              -10
 0  2.15650000d+01  1s2  2s2  2p6
 1  4.09640000d+01  1s2  2s2  2p5
 2  6.34600000d+01  1s2  2s2  2p4
 3  9.70800000d+01  1s2  2s2  2p3
 4  1.26220000d+02  1s2  2s2  2p2
 5  1.57930000d+02  1s2  2s2  2p1
 6  2.07280000d+02  1s2  2s2
 7  2.39100000d+02  1s2  2s1
 8  1.19583000d+03  1s2
 9  1.36221000d+03  1s1

C-----------------------------------------------------------------------
C
C Ionisation potentials : R L Kelly, J. Phys. Chem. Ref. Data,
C                                    vol. 16, Suppl. 1, 1987
C
C Update : Martin O'Mullane
C Date   : 10-09-2003
C
C-----------------------------------------------------------------------
```

Data lines (metastable resolved LS form):

Data lines	Format
ELEMENT, IZ0S (ICNCTV(k),k=0,|IZOS|-1)	1a16,1i5,30i3
for I=0,|IZ0S|-1	
IZ, EION(I+1),CFG(I+1)	i2,3x,1f16.8,1a120
for IM=1,ICNCTV(IZ+1)	
IM,EXM(I+1,IM),CFGM(I+1,IM)	2x,i3,1f16.8,1a120
endfor	
endfor	

Variable identification:

Name	Meaning	Comment
ELEMENT	element name	
IZ0S	signed nuclear charge	<0 => data lines include ionisation potential >0 => data lines do not include ionisation potential)
ICNCTV()	connection vector	number of designated metastables in each ionisation stage
IZ	ion charge	
EION()	ionisation potential	eV
CFG()	configuration string	note precise formatting with 5 character space allocation to each shell
IM	metastable index	
EXM(,)	metast/excit energy	relative to lowest metastable (the ground)of the stage (eV).
CFGM(,)	configuration string	precise formatting with 5 character space allocation to each shell.

Sample 2: **/home/adas/adas/adf00/ne_ls.dat**

```
.
neon              -10  2  2  4  3  4  2  2  1  2  1
 0     2.15967924d+01  1s2  2s2  2p6
    1  0.00000000d+00  1s2  2s2  2p6
    2  1.66470225d+01  1s2  2s2  2p5  3s1
 1     4.09699588d+01  1s2  2s2  2p5
    1  0.00000000d+00  1s2  2s2  2p5
    2  2.71747667d+01  1s1  2s2  2p4  3s1
 2     6.33840596d+01  1s2  2s2  2p4
    1  0.00000000d+00  1s2  2s2  2p4
    2  3.16459061d+00  1s2  2s2  2p4
    3  6.87319961d+00  1s2  2s2  2p4
    4  3.83810996d+01  1s2  2s2  2p3  3s1
 3     9.72103044d+01  1s2  2s2  2p3
    1  0.00000000d+00  1s2  2s2  2p3
    2  5.11466800d+00  1s2  2s2  2p3
    3  7.74145333d+00  1s2  2s2  2p3
 4     1.26230662d+02  1s2  2s2  2p2
    1  0.00000000d+00  1s2  2s2  2p2
    2  3.66216556d+00  1s2  2s2  2p2
    3  7.83077556d+00  1s2  2s2  2p2
    4  1.08614956d+01  1s2  2s1  2p3
 5     1.57822833d+02  1s2  2s2  2p1
    1  0.00000000d+00  1s2  2s2  2p1
    2  1.24081333d+01  1s2  2s1  2p2
 6     2.07270600d+02  1s2  2s2
```

```
    1  0.00000000d+00  1s2  2s2
    2  1.39122533d+01  1s2  2s1 2p1
 7     2.39096900d+02  1s2  2s1
    1  0.00000000d+00  1s2  2s1
 8     1.19582200d+03  1s2
    1  0.00000000d+00  1s2
    2  79.0507720d+02  1s1  2s1
 9     1.36219860d+03  1s1
    1  0.00000000d+00  1s1

C-----------------------------------------------------------------------
C
C Ionisation potentials : NIST  http://physics.nist.gov/
C                               PhysRefData/ASD/levels_form.html
C
C Excitation energies   : NIST  http://physics.nist.gov/
C                               PhysRefData/ASD/levels_form.html
C
C Update : Hugh Summers
C Date   : 05-01-2007
C
C-----------------------------------------------------------------------
```

A.2 *adf03*: recombination, ionisation and power parameters

Files contain sets of parameters of approximations to radiative recombination, dielectronic recombination, collisional ionisation, total radiated line power and specific line power coefficients for the ions of an element. These are sufficient to establish the ionisation state of an element in a thermal plasma. For each ion there are five coefficient sub-blocks and up to three different approximations may be used, namely, *case a*, *case b* and *case c*. For the ionisation part, *case b* is subdivided into *case ba*, *case bb*, and *case bb*. Approximations to the different coefficients may be in different cases in the one data set. The blocks are as follow:

Sub-block	Identifier	Case	Approximation reference
radiative recom.	CRRC	RRC#A	Abels van Maanaen
		RRC#B	this document: eqn. 3.15
		DRC#C	this document: eqn. 3.15
dielectronic recom.	CDRC	DRC#A	Summers and Dickson
		DRC#B	This document: eqn. 3.22
		DRC#C	This document: sec. 3.2.2
collisional ionis.	CCIO	CIO#A	Abels van Maanaen
		CIO#B	Summers and Dickson
		CIO#BA	This document: eqn. 3.5, fig.3.1
		CIO#BB	This document: eqn. 3.5, fig.3.1
		CIO#BC	This document: eqn. 3.5, fig.3.1
line radiated power	CPLT	PLT#A	Summers and Dickson
		PLT#B	Summers and Dickson
		PLT#C	Summers and Dickson
specific line power	CPLS	PLS#A	Summers and Dickson
		PLS#B	Summers and Dickson
		PLS#C	Summers and Dickson

Data mnemonic:
Data root: **/home/adas/adas/adf03/**

Last update: 27 November 2008
Utilising subroutines: **adas408.for**

Formatted files to adf03 specification:

Element	Prefix	Library	Comments	Quality
al,ar,b,be,c,cl,cr,f,fe,h,he,li, ne,ni,o	vm	atompars	JET ~ 1985	low
ar,c,cl,kr,li,n,ne,s,xe	mm	atompars	JET post 1989	low/medium
b,be,f,fe,ni	ms	atompars	JET post 1989	low/medium
ag,ar,kr,sn,w,xe	ca, ls, ic	baseline	heavy species 2009	low/medium/high

Notes: *mm* and *ms* use atomatic file preparation from adf04 specific ion files of *ss* or *mm* type with the code ADAS407. *vm* parameter compilations were prepared at JET in the 1985 period. They have been reformatted to *adf03* specification and are available for backward reproducibility of early work.

Data lines:

Data lines	Format
IZ0 , IZL , IZU , ISW1 , ISW2 , ISW3 , ADFID	3i5,10x,3i5
for iz=IZL,IZU	
———-	a10
ZR , IZD , IZI , IZT , IZS	5i5
CRRC , NRRC , ISRRC	1a11,i4,i5
(A) NZ , KSI	10x,2i5
(B) N0R , V0R , PHFACR, EDISPR, SCALER	10x,i5,4f10.3
CDRC , NDRC , ISDRC	1a11,i4,i5
for idrc=1,NDRC	
(A) DE , F , G , DN , MS	10x,3f10.3,2i5
(B) ITYPD, N0D, NCUT, V0D , PHFACD, CRFACD	10x,3i5,3f10.3
(B) EPSIJ, FIJ , EDISPD, SCALED	15x,4f10.3
endfor	
CCIO , NCIOS , NCIOR , ISCIO	1a11,i4,2i5
for icios=1,NCIOS	
(A) P , A , B , C , Q	10x,4f10.3,i5
(B, BA, BB, BC) ZETA, EION, CI	10x,3f10.3
endfor	
for icior=1,NCIOR	
(BC) WGHT, ENER, CR	10x,3f10.3
endfor	
CPLT , NPLT , ISPLT	1a11,i4,i5
for iplt=1,NPLT	
(A) DE , F , G , DN	10x,4f10.3
(B) DE , F , SPYLT	10x,3f10.3
endfor	
CPLS , NPLS , ISPLS , LINFO	1a11,i4,i5,f10.2
for ipls=1,NPLS	
(A) DE , F , G , DN	10x,4f10.3
(B) DE , F , SPYLS	10x,2f10.3
endfor	
endfor	
———-	a10

Variable identification:

Name	Meaning	Comment
IZ0	nuclear charge	
IZL	lowest included ion	
IZU	highest included ion	
ISW1	- not used -	
ISW2	- not used -	
ISW3	- not used -	
ADFID	ADAS data file type code	
IZR	recombining ion (rad. recom.)	
IZD	recombining ion (diel. recom.)	
IZI	ionising ion (coll. ionis.)	
IZT	radiating ion (total line power)	
IZS	radiating ion (specific line power)	
CRRC	radiative recom. code	
NRRC	- not used -	
ISRRC	- not used -	
CDRC	dielectronic recom. code	
NDRC	number of transitions following	
ISDRC	- not used -	
CCIO	collisional ionis. code	
NCIOS	number of shell values following	
NCIOR	number of reson. values following	
ISCIO	- not used -	
CPLT	total line power code	
NPLT	number of transitions following	
ISPLT	- not used -	
CPLS	specific line power code	
NPLS	- not used -	
ISPLS	- not used -	
LINFO	wavelength of specific line for naming purposes	

case (A) variables :

Name	Meaning	Comment
NZ	lowest accessible shell for rad. recom.	
KSI	number of electrons in shell	
DE	transition energy (eV)	
F	oscillator strength	
G	Gaunt factor	
DN	delta n for transition	
MS	Mertz switch (0=off, 1=on)	
P	shell ionisation potential (eV)	
	* N.B. The outer valence shell must occur first	
A	Lotz parameter	
B	Lotz parameter	
C	Lotz parameter	
Q	equivalent electrons in shell	

case (B, BA, BB, BC) variables :

Name	Meaning	Comment
N0R	lowest accessible princ. quantum shell for rad. recom.	
V0R	effective principal quantum number for shell	
PHFACR	phase space occupancy availability for shell	
EDISPR	energy adjustment in lowest shell rate coefficient (ryd)	
SCALER	multiplier for lowest shell rate coefficient	
ITYPD	Type of dielectronic transition	
N0D	lowest accessible princ. quantum shell for diel. recom.	
NCUT	cut-off princ. quantum shell in general program	
V0D	effective princ. quantum number for lowest access. shell	
PHFACD	phase space occupancy availability for lowest shell	
CRFACD	adjustment for Bethe corrections in general program	
EPSIJ	z-scaled parent transition energy (ryd)	
FIJ	oscillator strength for transition	
EDISPD	energy adjustment in Burgess general formula (ryd)	
SCALED	multiplier on Burgess general formula	
ZETA	number of equivalent electrons for shell	
EION	ionisation energy for shell (ryd)	
CI	multiplier for Burgess-Chidichimo rate for shell	
WGHT	weighting factor for excitation to resonance	
ENER	excitation energy for transition to resonance (ryd)	
CR	multiplier on excitation rate expresssion	
SPYLT	multiplier of Van Regemorter P factor in total power	
SPYLS	multiplier of Van Regemorter P factor in specific line	

Sample 1: **/home/adas/adas/adf03/atompars/vm#be.dat**

```
    4    0    3              1    0    0                                  ADF03
----------
    1    0    0    0    0
     RRC#A    0    0
              2    1
     DRC#A    1    0
               0.0       0.0       0.0      1    1
     CIO#A    2    0    0
               9.320     4.000     0.700     0.500    2
              115.00     4.400     0.600     0.600    2
     PLT#A    2    0
               5.277     1.360     0.032    0
               7.462     0.021     0.157    1
     PLS#A    1    0   2349.30
               5.277     1.360     0.026    0
----------
    2    1    1    1    1
     RRC#A    0    0
              2    0
     DRC#A    2    0
               3.939     0.499     0.478    0    0
              11.864     0.096     0.123    1    1
     CIO#A    2    0    0
              18.200     4.400     0.000     0.000    1
             125.000     4.500     0.400     0.600    2
     PLT#A    2    0
               3.859     0.507     0.534    0
              11.963     0.080     0.135    1
     PLS#A    1    0   3131.50
               3.959     0.507     0.478    0
----------
    3    2    2    2    2
     RRC#A    0    0
              1    1
     DRC#A    2    0
             124.191     0.592     0.088    1    0
             140.719     0.152     0.073    1    0
     CIO#A    1    0    0
             154.000     4.500     0.300     0.600    2
     PLT#A    2    0
             123.662     0.547     0.271    1
             140.382     0.149     0.225    1
     PLS#A    1    0    100.30
             123.662     0.547     0.088    1
----------
    4    3    3    3    3
     RRC#A    0    0
              1    0
     DRC#A    2    0
             163.258     0.415     0.201    1    0
             193.490     0.079     0.252    1    0
     CIO#A    1    0    0
             218.000     4.500     0.000     0.000    1
     PLT#A    2    0
```

```
           163.367     0.414     0.258    1
           193.611     0.079     0.324    1
   PLS#A    1    0    75.900
           163.367     0.414     0.201    1
------------------------------------------------------------------------------------------
  Data source : Abels-van Maanen  - A package for non-coronal impurity
                                    data  JET-DN-T(85)28 (pp13-15).


  Alterations : (??/??/89) Janeschitz -
                                  - Extension to include specific
                                    line parameters.
                (13/12/90) Summers    - Reformatted to ADAS data format
                                    convention  ADF03.
                                    Reassembled and reassessed for
                                    consistency with van Maanen.
                                    NB. Mertz correction is not
                                    activated on H-like and He-like
                                    unlike Abels-van Maanen.
                (27/ 2/91) Summers    - DRC, PLT & PLS in table below

  +-------+------------------------------+-------------+--------------+
  + Ion   + Specific ion data source     + Analysing   + Matching Te  +
  +       +                              + code for    + for PLT Gaunt+
  +       +                              + parameters  +    factors   +
  +-------+------------------------------+-------------+--------------+
  + Be+ 0 + JETSHP.BELIKE.DATA(FBBH91BE) + POWERFIT    +    0.86eV    +
  +       +                              +             +-473% @ 86eV  +
  +       +                              +             + +32% @ 0.43eV+
  +       +                              +             +              +
  +-------+------------------------------+-------------+--------------+
  + Be+ 1 + JETSHP.BELIKE.DATA(WJD91BE)  + POWERFIT    +    6.9eV     +
  +       +                              +             +-363% @ 345eV +
  +       +                              +             + +35% @ 1.7eV +
  +       + JETXLE.COPDT#BE.DATA(SS#BE1) + EXP90       +              +
  +-------+------------------------------+-------------+--------------+
  + Be+ 2 + JETSHP.BELIKE.DATA(HPS90BE)  + POWERFIT    +   15.5eV     +
  +       +                              +             + -90% @ 388eV +
  +       +                              +             +-120% @ 1.6eV +
  +       + JETXLE.COPDT#BE.DATA(SS#BE2) + EXP90       +              +
  +-------+------------------------------+-------------+--------------+
  + Be+ 3 + JETSHP.BELIKE.DATA(HPS90BE)  + POWERFIT    +   30.0eV     +
  +       +                              +             + -60% @ 300eV +
  +       +                              +             + +8%  @ 10eV  +
  +       + JETXLE.COPDT#BE.DATA(SS#BE3) + EXP90       +              +
  +-------+------------------------------+-------------+--------------+


  Formatting  : Defined in member  ADF03


  Usage       : Formatted for access by subroutine  NCRATNT

------------------------------------------------------------------------------------------
```

A.3 *adf04*: resolved specific ion data collections

Provides all required energy level and rate coefficient data for specified low levels of an ion. The data set is complete for a low level population calculation. Specific level selective free electron recombination, ionisation and charge exchange recombination are included. Formatting conventions and variable storage are given below. Note that current preferred data for important elements are grouped into isonuclear libraries of the form **adas#**.

Data mnemonic: specific ion file
Data root: **/home/adas/adas/adf04/**
Last update: 17 March 2007
Utilising subroutines: **xxdata_04.for**

Formatted files to adf04 specification:

Sequence	Members	Library	Comments	Resol.	Quality
H-like	H,He,Li,Be,B,C,N,O,F,Ne	copha#h	Anderson (n=1,2,3,4,5)	n & nl	high
He-like	Li,Be,B,C,N,O,F,Ne	copha#he	Anderson (n=1,2,3,4,5)	l	high
Li-like	Be,C,N,O,Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	coppm#li	McWhirter (n=2,3,4,5)	j	high
Be-like	C,N,O,Ne,Mg,Al,Si,S,Ar, Ca,Ti,Fe,Ni	copjl#be	Lang (n=2,3)	j	high
C-like	Ca,Fe,Mg,Ne,O,S,Si	clike	Monsignori-Fossi(n=2)	j	assessed
N-like	Ar,Ca,Mg,S,Si	nlike	Landini(n=2)	j	assessed
O-like	Ne,Mg,Al,Si,S,Ar, Ca,Ti,Fe,Ni	olike	Lang/Summers (n=2)	j	best available

Notes:

1. **copha#h** directory has four groups of members (a) prefix **bn#h** => n–shell, high temperature set (b) prefix **bn#l**=> n-shell, low temperature set (c) prefix **hah** => nl-shell, high temperature set (d) prefix **hal**=> nl-shell, low temperature set.

2. **copha#he** directory members have prefix **sm#**=¿initial data based on Sampson calculations. Improved data then introduced where available.

3. There are many specific ion files individually assessed for ions of light elements and ions of special fusion interest in the ADAS database. These are in iso-electronic sequence sub-directories with members named by date and originator as
/<seq>like/<seq>like_<source><year><el>.dat where <seq> is the ioslectronic sequence symbol, ¡source¿ is the orginators initials, <year> a two digit year number and <el> the element symbol.

4. Lang has prepared a detailed document on adf04 files for matching to SOHO/CDS specific needs as detailed by the Blue Book. This is available in the datastatus section of ADAS documentation on the web.

Sequence	Members	Library	Comments	Resol.	Quality
Li-like	Be,C,N,O,Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copsm#li	Sampson/Zhang(n=2,3,4,5)	j	medium
Be-like	C,N,O,Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copsm#be	Sampson/Zhang(n=2)	j	medium
B-like	C,N,O,Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copsm#b	Sampson/Zhang(n=2,3)	j	medium
C-like	N,O,Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copsm#c	Sampson/Zhang(n=2,3)	j	medium
F-like	Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copsm#f	Sampson/Zhang(n=2,3)	j	medium
Ne-like	Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copsm#ne	Sampson/Zhang(n=3,4)	j	medium
Na-like	Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copsm#na	Sampson/Zhang(n=3,4,5)	j	medium

Sequence	Members	Library	Comments	Resol.	Quality
Li-like	Be,C,N,O,Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copss#li	SS/dip i.p.(n=2,3,4,5)	j	low
Be-like	C,N,O,Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copss#be	SS/dip i.p.(n=2)	j	low
B-like	C,N,O,Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copss#b	SS/dip i.p.(n=2,3)	j	low
C-like	N,O,Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copss#c	SS/dip i.p.(n=2,3)	j	low
F-like	Ne,Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copss#f	SS/dip i.p.(n=2,3)	j	low
Ne-like	Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copss#ne	SS/dip i.p.(n=3,4)	j	low
Na-like	Mg,Al,Si,S,Ar Ca,Ti,Fe,Ni	copss#na	SS/dip i.p.(n=3,4,5)	j	low

Sequence	Members	Library	Comments	Resol.	Quality
H isonuc.	all ions	adas#1	OMullane	ls & n	preferred
He isonuc	all ions	adas#2	OMullane	ic & ls	preferred
Li isonuc	all ions	adas#3	Ballance	ls, bd-n	preferred
C isonuc.	all ions	adas#6	OMullane	ic & ls	preferred
N isonuc.	all ions	adas#7	Dux	ic & ls	preferred
O isonuc.	all ions	adas#8	Broooks	ic & ls	preferred
Ne isonuc	all ions	adas#10	OMullane	ic & ls	preferred
Ar isonu	Ar+18 only	adas#18	Whiteford	ic	preferred

Sequence	Members	Library	Comments	Resol.	Quality
H isonuc.	all ions	copmm#1	Cowan/Born(n=1-5)	j & ls	low
He isonuc.	all ions	copmm#2	Cowan/Born(n=1-5)	j & ls	low
Li isonuc.	all ions	copmm#3	Cowan/Born(n=2,3)	ls	low
Be isonuc.	all ions	copmm#4	Cowan/Born(n=2,3)	j & ls	low
B isonuc.	all ions	copmm#5	Cowan/Born(n=2,3)	j & ls	low
C isonuc.	all ions	copmm#6	Cowan/Born(n=2,3)	j & ls	low
N isonuc.	all ions	copmm#7	Cowan/Born(n=2,3)	j & ls	low
O isonuc.	all ions	copmm#8	Cowan/Born(n=2,3)	j & ls	low
F isonuc.	all ions	copmm#9	Cowan/Born(n=2,3)	j & ls	low
Ne isonuc.	all ions	copmm#10	Cowan/Born(n=2,3)	j & ls	low
Si isonuc.	all ions	copmm#16	Cowan/Born(n=2,3)	j & ls	low
Cl isonuc.	all ions	copmm#17	Cowan/Born(n=2,3)	j & ls	low
Ar isonuc.	all ions	copmm#18	Cowan/Born(n=2,3)	j & ls	low
Fe isonuc.	all ions	copmm#26	Cowan/Born(n=2,3)	j & ls	low
Ni isonuc.	all ions	copmm#28	Cowan/Born(n=2,3)	j & ls	low
Kr isonuc.	all ions	copmm#36	Cowan/Born(n=2,3)	j & ls	low
Xe isonuc.	all ions	copmm#54	Cowan/Born(n=2,3)	j & ls	low

Notes:

1. adas sub-libraries contain the currently preferred data for the ions of the element. The prefix ls indicates the LS-coupled data complete with all recombination contributions for full generalised-collisional-radiative population calculations. The prefix ic indicates intermediate j-resolved data with spectroscopic quality energies. The latter data sets do not have the recombination contributions. The prefix n indicates n-shell bundled data and is relevant only to the hydorgen-like stage. When a change of preferred data is made, the formerly preferred data is moved back into isoelectronic data collections with an advisory note as to when they were the preferred data. 24 Oct. 1999 revision includes new datasets in **/adas#7** and **/adas#8** from the GCR project and updates to **/adas#2** and **/adas#6**.
2. **copss** files contain dipole collision rate coefficients only. They are often used for supplementation of higher quality data sets which lack dipole allowed transitions between higher levels.
3. **copmm** files contain dipole and non-dipole non-spin change collision rate coefficients only. Exchange collisions are not present.
4. **copmm** & **copss** files are useful for initial survey and radiated power assessment in ADAS series 4 codes.
5. **copss** & **copmm** files are generally created with Eissner form configuration notation for automatic initial set up of metastable resolved collisional-radiative recombination/ionisation calculation with ADAS series 2 codes.

Data lines:

Data lines	Format
SYM , IZ , IZ0 , IZ1 , STRG1	1a3,i2,2i10,1a75
until IND = -1	i5, 1a95
IND , CFG, IS , IL , C8, STRG2	i5,1x,1a18,1x,i1,1x,i1,1a8,1a56
-1 , STRG3	
ZEFF , ITYP , STRG4	f5.1,i5,6x,1a112
until INDU = -1 and INDL = -1	
until INDU = -1	
CCODE, INDU , INDL , STRG5	i1a1,1i3,i4,1a128
-1	4x, 1i4
-1 -1	2i4
C+++ERROR specification start++++++	1a35
C	
C TCLASS1	1a2,1a
C	
C IL-IU ERROR	1a2,1a,-,1a, -,fm.n
C	
C TCLASS2	1a2,1a
C	
C IL-IU ERROR	1a2,1a,-,1a, -,fm.n
C	
C+++ERROR specification end++++++++	1a35

Variable identification:

Name	Meaning	Comment
ELEMENT	element symbol	in form ##+
IZ	charge of the ion	
IZ0	nuclear charge	
IZ1	ion charge + 1	
STRG1	ion. pot. string	*fword1(cword1)fword2(cword2)*$\cdots$ where the *fword* are fixed point decimal numbers and *cword* are character strings fword1 = BWNO = BWNOA(1) ≡ ion. pot. (cm-1) fwordi = BWNOA(I) ≡ ion. pot. (cm-1) of lowest level relative to the ith parent cwordi=MLTP,LP ≡ (2Sp+1)Lp for ith parent in LS coupling cwordi=MLTP,LP,XJ ≡ (2Sp+1)Lp Jp for ith parent in IC coupling

Sample 1: **/home/adas/adas/adf04/ne.dat**

A.4 *adf07*: direct resolved electron impact ionis. data collections

Provides electron impact ionisation rate coefficients for ions optionally with resolution into metastable initial and final states. Formatting conventions and variable storage are given below.

Data mnemonic: szd
Data root: **/home/adas/adas/adf07/**
Last update: 15 January 2009
Utilising subroutines: **xxdata_07.for**, **read_adf07.pro**

Formatted files to adf07 specification:

Element	Members	Library	Comments	Resol.	Quality
hydrogen	h,h0	szd93#h	CLM normalised, S&H split	resolved	high
helium	he,he0,he1	szd93#he	CLM normalised, S&H split	resolved	high
lithium	li0-li2	szd02#li	Loch, Colgan	resolved	high
lithium	li	szd93#li	CLM normalised	unresolved	high
beryllium	be,be0-be3	szd93#be	CLM normalised, S&H split	resolved	medium
boron	b	szd93#b	CLM normalised	unresolved	high
carbon	c,c0-c5	szd93#c	CLM normalised, S&H split	resolved	medium
nitrogen	n	szd93#n	CLM normalised	unresolved	high
nitrogen	n,n0-n6	szd96#n	CLM normalised, S&H split	resolved	medium
oxygen	o,o0-o7	szd93#o	CLM normalised, S&H split	resolved	medium
fluorine	f	szd93#f	CLM normalised	unresolved	high
neon	ne	szd93#ne	CLM normalised	unresolved	high
argon	ar1	szd97#ar	Griffin, normalised	resolved	high
chromium	cr,cr0,cr1	szd93#cr	Griffin, normalised	resolved	high
iron	fe	szd93#fe	A&R	unresolved	medium
molybdenum	mo,mo0,mo1	szd93#mo	Griffin, normalised	resolved	high
argon	ca#ar	sdl08#18	Loch-cadw, baseline	*ca* grds. only	medium
krypton	ca#kr	sdl08#36	Loch-cadw, baseline	*ca* grds. only	medium
xenon	ca#xe	sdl08#54	Loch-cadw, baseline	*ca* grds. only	medium
silver	ca#ag	sdl08#47	Loch-cadw, baseline	*ca* grds. only	medium
tin	ca#sn	sdl08#50	Loch-cadw, baseline	*ca* grds. only	medium
tungsten	ca#w	sdl08#74	Loch-cadw, baseline	*ca* grds. only	medium

Notes:

1. The element member contains a preferred compilation for all the ions of the element. The ion members contain just the individual ion resolved data.
2. CLM normalised refers to the Culham Laboratory Reports CLM-R216- & CLM-R270 for the total stage to stage coefficients.
3. S&H refers to splitting into metastable resolved ionisation coefficients using the method of Summers & Hooper. Results are normalised to the stage to stage total.
4. Griffin refers to metastable resolved data extracted from the detailed adf23 data sets prepared by Griffin, Pindzola or Badnell. This data is also normalised as specified in the dataset comments. Griffin data will eventually replace all the S&H split data.
5. A&R refers to Arnaud & Raymond (1992) Astrophys. J. 398, 394.
6. Heavy element baseline data in the *cadw* approximation, has in general both direct and excitation/autoionisation parts and includes both ground state and excited state ionisation. Such complete data of format *adf23* is totalled (including direct and excitation/auotionisation) from the ground state of each ion of the elements and returned in the present format for the convenience of users. .

Data lines:

Data lines	Format
NSEL, TEXT	i5,4x,'/',1a38,'/'
for ISEL= 1 ,NSEL	
SYMB, IZ, SYMB IZ1, NTE, BWNO, METI, METF, ISEL	c2,'+',i2,'/',c2,'+',i2'/',i5,'/',7x,f18,9x,i2,9x,i2,7x,i3
(TEV(IT), IT=1,NTE)	6e10.3
(SZD(IT), IT=1,NTE)	6e10.3
endfor	

Variable identification:

Name	Meaning	Comment
NSEL	number of reactions available	
TEXT	information	
SYMB	element chemical symbol	
IZ	initial ion charge	
IZ1	final ion charge	
NTE	number of temperatures	
BWNO	effective ionisation potential (cm-1)	
METI	initial state metastable index	
METF	final state metastable index	
ISEL	transition index	
TEV()	electron temperatures (eV)	
SZD()	zero density ionis. coefft. (cm**3 sec-1)	

Sample 1: **/home/adas/adas/adf07/sll08#74/ca#w.dat**

```
   74 / tungsten      ionisation rate coefficients /ca/                    adf07
w + 0/w + 1/    9/i.p. =            61360.6/icode =  1/fcode =  1/isel =   1
 4.309E-01 8.618E-01 1.724E+00 4.309E+00 8.618E+00 1.724E+01
 4.309E+01 8.618E+01 1.724E+02
 1.547E-15 1.605E-11 1.994E-09 4.554E-08 1.469E-07 2.858E-07
 4.401E-07 4.840E-07 4.745E-07
w + 1/w + 2/   12/i.p. =           119430.5/icode =  1/fcode =  1/isel =   2
 6.894E-01 1.724E+00 3.447E+00 6.894E+00 1.724E+01 3.447E+01
 6.894E+01 1.724E+02 3.447E+02 6.894E+02 1.724E+03 3.447E+03
 1.359E-17 1.052E-11 1.221E-09 1.524E-08 8.350E-08 1.577E-07
 2.111E-07 2.340E-07 2.235E-07 1.989E-07 1.583E-07 1.288E-07
w + 2/w + 3/   12/i.p. =           200548.8/icode =  1/fcode =  1/isel =   3
 1.551E+00 3.878E+00 7.756E+00 1.551E+01 3.878E+01 7.756E+01
 1.551E+02 3.878E+02 7.756E+02 1.551E+03 3.878E+03 7.756E+03
 4.098E-15 9.602E-11 3.474E-09 2.458E-08 8.605E-08 1.214E-07
 1.367E-07 1.317E-07 1.160E-07 9.668E-08 7.199E-08 5.617E-08
w + 3/w + 4/   12/i.p. =           297980.3/icode =  1/fcode =  1/isel =   4
 2.758E+00 6.894E+00 1.379E+01 2.758E+01 6.894E+01 1.379E+02
 2.758E+02 6.894E+02 1.379E+03 2.758E+03 6.894E+03 1.379E+04
 7.211E-14 4.652E-10 8.480E-09 3.590E-08 7.426E-08 8.379E-08
 8.164E-08 6.729E-08 5.379E-08 4.115E-08 2.788E-08 2.072E-08
w + 4/w + 5/   12/i.p. =           403546.1/icode =  1/fcode =  1/isel =   5
 2.482E+01 6.205E+01 1.241E+02 2.482E+02 6.205E+02 1.241E+03
 2.482E+03 6.205E+03 1.241E+04 2.482E+04 6.205E+04 1.241E+05
 5.536E-09 2.273E-08 3.157E-08 3.471E-08 3.209E-08 2.734E-08
 2.200E-08 1.575E-08 1.197E-08 9.001E-09 6.115E-09 4.551E-09
```

```
          .   .   .

w +71/w +72/   12/i.p. =        158739837.8/icode =  1/fcode =  1/isel =  72
 8.962E+02 2.232E+03 4.464E+03 8.962E+03 2.232E+04 4.464E+04
 8.962E+04 2.232E+05 4.464E+05 8.962E+05 2.232E+06 4.464E+06
 5.779E-23 4.575E-17 4.807E-15 5.263E-14 2.256E-13 3.681E-13
 4.591E-13 4.779E-13 4.387E-13 3.760E-13 2.886E-13 2.291E-13
w +72/w +73/   12/i.p. =        638408392.0/icode =  1/fcode =  1/isel =  73
 9.221E+02 2.292E+03 4.593E+03 9.221E+03 2.292E+04 4.593E+04
 9.221E+04 2.292E+05 4.593E+05 9.221E+05 2.292E+06 4.593E+06
 5.566E-52 2.886E-29 1.441E-21 1.128E-17 2.618E-15 1.740E-14
 4.535E-14 7.576E-14 8.350E-14 8.012E-14 6.714E-14 5.563E-14
w +73/w +74/   12/i.p. =        649757158.2/icode =  1/fcode =  1/isel =  74
 9.652E+02 2.422E+03 4.843E+03 9.652E+03 2.422E+04 4.843E+04
 9.652E+04 2.422E+05 4.843E+05 9.652E+05 2.422E+06 4.843E+06
 2.928E-51 4.956E-29 1.272E-21 6.927E-18 1.440E-15 8.944E-15
 2.235E-14 3.700E-14 4.056E-14 3.873E-14 3.240E-14 2.675E-14
c-----------------------------------------------------------------------------------------------------
c
c  Processed from adf23 datasets
c
c  Code     : process_adf23_adf07.for
c  Author   : Hugh Summers
c  Date     : 22-05-2008
c
c  Sources  :
c
c  isel   filename
c  ----   --------
c    1    /home/summers/adas_dev/adas/adf23/sdl08#74/ca#w0.dat
c    2    /home/summers/adas_dev/adas/adf23/sdl08#74/ca#w1.dat
c    3    /home/summers/adas_dev/adas/adf23/sdl08#74/ca#w2.dat
c    4    /home/summers/adas_dev/adas/adf23/sdl08#74/ca#w3.dat
c    5    /home/summers/adas_dev/adas/adf23/sdl08#74/ca#w4.dat

          .   .   .

c   72    /home/summers/adas_dev/adas/adf23/sdl08#74/ca#w71.dat
c   73    /home/summers/adas_dev/adas/adf23/sdl08#74/ca#w72.dat
c   74    /home/summers/adas_dev/adas/adf23/sdl08#74/ca#w73.dat
c
c
c  Details:
c
c  isel  init.  im_i  imeta_i  cfg_i                          final  im_f  imeta_f  cfg_f
c  ----  -----  ----  -------  -----------------------------  -----  ----  -------  ---------------
c    1   w + 0    1       1     5p6 5d4 6s2                   w + 1    1       1     5p6 5d4 6s1
c    2   w + 1    1       1     5s2 5p6 5d4 6s1               w + 2    1       1     5s2 5p6 5d4
c    3   w + 2    1       1     5s2 5p6 5d4                   w + 3    1       1     5s2 5p6 5d3
c    4   w + 3    1       1     5s2 5p6 5d3                   w + 4    1       1     5s2 5p6 5d2
c    5   w + 4    1       1     5s2 5p6 5d2                   w + 5    1       1     5s2 5p6 5d1

          .   .   .

c   72   w +71    1       1     1s2 2s1                       w +72    1       1     1s2
c   73   w +72    1       1     1s2                           w +73    1       1     1s1
```

```
c   74   w +73    1       1     1s1                          w +74    1       1     1s0
c
c  isel  init.  im_i  imeta_i  (stat.wt.-1)/2                final  im_f  imeta_f  (stat.wt.-1)/2
c  ----  -----  ----  -------  --------------                -----  ----  -------  --------------
c    1   w + 0    1       1         104.5                    w + 1    1       1          209.5
c    2   w + 1    1       1         209.5                    w + 2    1       1          104.5
c    3   w + 2    1       1         104.5                    w + 3    1       1           59.5
c    4   w + 3    1       1          59.5                    w + 4    1       1           22.0
c    5   w + 4    1       1          22.0                    w + 5    1       1            4.5

          .   .   .

c   72   w +71    1       1           0.5                    w +72    1       1            0.0
c   73   w +72    1       1           0.0                    w +73    1       1            0.5
c   74   w +73    1       1           0.5                    w +74    1       1            0.0
c
c-------------------------------------------------------------------------------------------------
```

A.5 *adf08*: direct resolved radiative recombination coefficients

Provides resolved radiative recombination coefficient data. Formatting conventions and variable storage are given below.

Data mnemonic: rrc
Data root: **/home/adas/adas/adf08/**
Last update: 15 January 2009
Utilising subroutines: **xxdata_08.for**, **read_adf08.pro**

Formatted files to adf08 specification:

Recombining seq.	Members	Library	Comments	Resol.	Quality
various		radrec		*ls*	high
bare nucl.	b, be, c, h, he, o	rrc93##		*ls*	medium
h-like.	b, be,c, he, o	rrc93#h		*ls*	medium
he-like	b, be, c, o	rrc93#he		*ls*	medium
li-like	b, be, c, o	rrc93#li		*ls*	medium
be-like	b, c, o	rrc93#be		*ls*	medium
b-like	c, o	rrc93#b		*ls*	medium
c-like	o	rrc93#c		*ls*	medium
n-like	o	rrc93#n		*ls*	medium
ti-like	cr	rrc93#ti		*ls*	medium
v-like	cr	rrc93#v		*ls*	medium
zr-like	mo	rrc93#zr		*ls*	medium
nb-like	mo	rrc93#nb		*ls*	medium
bare nucl.	c	rrc96##		*ls*	high
h-like.	c	rrc96#h		*ls*	medium
he-like	c	rrc96#he		*ls*	medium
li-like	c	rrc96#li		*ls*	medium
be-like	c	rrc96#be		*ls*	medium
b-like	c	rrc96#b		*ls*	medium
bare nucl.	n, o, ne	rrc98##		*ls*	high
h-like.	n, o, ne	rrc98#h		*ls*	medium
he-like	n, o, ne	rrc98#he		*ls*	medium
li-like	n, o, ne	rrc98#li		*ls*	medium
be-like	n, o, ne	rrc98#be		*ls*	medium
b-like	n, o, ne	rrc98#b		*ls*	medium
c-like	n, o, ne	rrc98#c		*ls*	medium
n-like	o, ne	rrc98#n		*ls*	medium
o-like	ne	rrc98#o		*ls*	medium
f-like	ne	rrc98#f		*ls*	medium

Notes:

1. In the individual data set names, the recombining ion symbol is followed by 'ls' to indicate LS resolution.
2. For proton or deuteron recombination, the 'ls' postfix on the dataset names is followed by 'h' or 'l' to denote high and low temperature ranges.
3. 24 Oct. 1994 revision includes replacement of /rrc98#b/rrc98#b_o3ls.dat. .

Data lines:

Data lines	Format
seq =’SEQ’ , nucchg =IZ0 , type =’CTYPE’ , ADF08	1a80: keywords - a2,i,a2,a5
<blank line>	1a80
c10 , bwnf = BWNF nprf = NLVL_F	1a80: a10,keywords -f12,i3
[c10 has form ’aa+nn aaaa’ where aa = elem. symb., nn= final ion charge and aaaa = conf/term/levl if CTYPE=ca/ls/ic] [characters are left justified and numbers right justified]	
<underlines>	1a80
<descriptive headers>	1a80
<underlines>	1a80
for i=1,NLVL_F	
IA_F(i) , CODE_F(i) , CSTRGA_F(i) , cnn , WA_F(i)	i5,a1,a,a,f13
[cnn has the form ’(ISA_F(i))ILA_F(i)(XJA_F(i))’ if CTYPE=ls/ic] and cnn has the form ’(XJA_F(i))’ if CTYPE=ca]	
endfor	
<blank line>	1a80
c10 , bwni = BWNI c4 = NLVL_I	1a80: a10,keywords -f12,i3
[c10 has form ’aa+nn aaaa’ where aa = elem. symb., nn= initial ion charge and aaaa = conf/term/levl if CTYPE=ca/ls/ic]	
[c4 has form ’aaaa’ where aaaa = ncfg/ntrm/nlvl if CTYPE=ca/ls/ic]	
<underlines>	1a80
<descriptive headers>	1a80
<underlines>	1a80
for i=1,NLVL_I	
IA_I(i) , CODE_I(i) , CSTRGA_I(i) , cnn , WA_I(i)	i5,a1,a,a,f13
[cnn has the form ’(ISA_I(i))ILA_I(i)(XJA_I(i))’ if CTYPE=ls/ic and cnn has the form ’(XJA_I(i))’ if CTYPE=ca]	
endfor	
<blank line>	1a80
<underlines>	1a80
do i=1,NMETI	
<underlines>	1a80
<descriptive headers>	1a80
<underlines>	1a80
meti*= IMET(i)	1a80: keywords - i
te= (TEA_ION(i,it),it=1,NTE)	1a256:keyword - d9.2,21d10.2
<underlines>	1a80
for j = 1 , NLVN_F	
INDF , (QRED_ION(i,indf,it),it=1,NTE)	i5,18x,22d10.2
endfor	
<blank line>	1a80
c20	a20

Variable identification:

Name	Meaning	Comment
SEQ	iso-electronic sequence	two characters, blank fill to right
IZ0	nuclear charge	
CTYPE	ca, ls or ic denoting resolution	two characters
BWNF	ionisation potential of final state ion (cm-1)	
NLVL_F	no. of final ion configs, terms or levels	depending on resolution
IA_F()	index of final states	
CODE_F()	code for metastables of final state ion	* indicates a metastable
CSTRGA_F()	configuration string of final states	standard notation
ISA_F()	multiplicity	only for ls and ic resolution
ILA_F()	total orbital quantum number	only for ls and ic resolution
XJA_F()	(stat. wt.-1)/2 or J	J only for ic resolution
BWNI	ionisation potential of initial state ion (cm-1)	
NLVL_I	no. of initial ion configs, terms or levels	depending on resolution
IA_I()	index of initial states	
CODE_I()	code for metastables of initial state ion	* indicates a metastable
CSTRGA_I()	configuration string of initial states	standard notation
ISA_I()	multiplicity	only for ls and ic resolution
ILA_I()	total orbital quantum number	only for ls and ic resolution
XJA_I()	(stat. wt.-1)/2 or J	J only for ic resolution
NMETI	no. of metastables of initial ion	
IMETI()	pointers to metastables in initial state index	
TEA_ION(,)	elec. temperatures (K) of ionis. coefft table	1st dim: initial metastable index, 2nd dim: te index
NTE	number of elec. temperatures	
INDF	current final state metastable index	
QRED_ION(,,)	scaled ionis. coeffts (cm3 s-1) table	1st dim: initial metastable index, 2nd dim: final state index 3rd dim: te index

Sample 1: **/home/adas/adas/adf08/rrc96#b_c1ls.dat**

```
SEQ='C '     NUCCHG= 6

   PARENT TERM INDEXING            BWNP=    196622.4   NPRNTI= 2
   --------------------
   INDP    CODE                 S L   WI        WNP
   ----    ----                 - -   --  ----------
     1     2S2 2P1             (2)1( 2.5)        0.0
     2     2S1 2P2             (4)1( 5.5)    42993.5

   LS RESOLVED TERM INDEXING       BWNR=     90878.0    NTRM= 64
   -------------------------
   INDX    CODE                 S L   WJ        WNR
   ----    ----                 - -   --  ----------
     1     2S2 2P2             (3)1( 4.0)        0.0  {1}2.00  {2}1.33
     2     2S2 2P2             (1)2( 2.0)    10163.0  {1}2.00
     3     2S2 2P2             (1)0( 0.0)    21618.4  {1}2.00
     4     2S1 2P3             (5)0( 2.0)    33705.6  {2}3.00
     5     2S2 2P1 3S1         (3)1( 4.0)    60343.3  {1}1.00

             .   .   .

    60     2P4                 (1)0( 0.0)   211514.3  {X}
    61     2S1 2P2 3S1         (5)1( 7.0)    97858.0  {2}1.00
    62     2S1 2P2 3D1         (5)3(17.0)   115887.3  {2}1.00
```

```
   63     2S1 2P2 3D1         (5)1( 7.0)   115909.5  {2}1.00
   64     2S1 2P2 3D1         (5)2(12.0)   116178.2  {2}1.00



---------------------------------
PRTI= 1  TRMPRT= (2P)  SPNPRT= 2

  INDX TE=  1.00D+04  1.25D+04  2.50D+04  3.75D+04  5.00D+04  7.50D+04  1.00D+05  1.25D+05
    1       2.12D-13  1.95D-13  1.54D-13  1.36D-13  1.24D-13  1.08D-13  9.69D-14  8.79D-14
    2       1.04D-13  9.54D-14  7.50D-14  6.58D-14  5.99D-14  5.21D-14  4.67D-14  4.23D-14
    3       1.79D-14  1.64D-14  1.27D-14  1.11D-14  1.00D-14  8.64D-15  7.70D-15  6.97D-15
    5       1.90D-15  1.86D-15  1.84D-15  1.86D-15  1.88D-15  1.90D-15  1.91D-15  1.90D-15
              .   .   .

   50       6.00D-16  4.91D-16  2.43D-16  1.53D-16  1.07D-16  6.41D-17  4.37D-17  3.23D-17
   51       1.53D-16  1.25D-16  6.21D-17  3.91D-17  2.76D-17  1.64D-17  1.12D-17  8.29D-18
   52       1.09D-16  8.92D-17  4.42D-17  2.78D-17  1.96D-17  1.17D-17  7.99D-18  5.90D-18
   53       1.96D-16  1.60D-16  7.94D-17  5.00D-17  3.52D-17  2.10D-17  1.44D-17  1.06D-17
   54       3.26D-16  2.67D-16  1.32D-16  8.32D-17  5.87D-17  3.50D-17  2.39D-17  1.77D-17


---------------------------------
PRTI= 2  TRMPRT= (4P)  SPNPRT= 4

  INDX TE=  1.00D+04  1.25D+04  2.50D+04  3.75D+04  5.00D+04  7.50D+04  1.00D+05  1.25D+05
    1       1.42D-14  1.32D-14  1.13D-14  1.08D-14  1.06D-14  1.06D-14  1.07D-14  1.06D-14
    4       9.51D-14  8.79D-14  7.05D-14  6.26D-14  5.75D-14  5.05D-14  4.55D-14  4.13D-14
    7       1.30D-13  1.20D-13  9.37D-14  8.22D-14  7.49D-14  6.53D-14  5.86D-14  5.33D-14
   14       6.68D-14  6.10D-14  4.67D-14  4.03D-14  3.63D-14  3.12D-14  2.77D-14  2.50D-14
   56       5.19D-15  4.47D-15  2.71D-15  1.96D-15  1.54D-15  1.07D-15  8.12D-16  6.52D-16
   61       2.87D-15  2.77D-15  2.61D-15  2.59D-15  2.60D-15  2.60D-15  2.60D-15  2.57D-15
   62       4.51D-14  4.08D-14  2.97D-14  2.41D-14  2.05D-14  1.59D-14  1.30D-14  1.09D-14
   63       1.93D-14  1.74D-14  1.27D-14  1.03D-14  8.76D-15  6.78D-15  5.53D-15  4.66D-15
   64       3.10D-14  2.80D-14  2.02D-14  1.64D-14  1.39D-14  1.08D-14  8.77D-15  7.38D-15
```

A.6 *adf09*: state selective dielectronic recombination coefficients

Provides state-selective dielectronic recombination coefficient data in *ls* and *ic* resolution. The data is resolved by initial and final parent metastable and includes capture to *ls*- or *ic*- resolved low levels, *bundle-nl* shells and *bundle-n* shells to very high n. Data sets are generally distinguished according to parent principal quantum shell transition ($n \rightarrow n\prime$) and possibly by captured principal quantum shell ($n\prime\prime$) as $nn\prime n\prime\prime$. Formatting conventions and variable storage are given below.

Data mnemonic: drc
Data root: **/home/adas/adas/adf09/**
Last update: 23 January 2009
Utilising subroutines: **xxdata_09.for**, **read_adf09.pro**

Formatted files to adf09 specification:

Recombining seq.	Members	Library	$n \rightarrow n\prime$	Resol.	Quality
H-like.	Ar,B,Be,C,Fe,He,Li, Mg,O,Sn,Y	nrb93#h	1-2	*ls*	medium
He-like	Ar,B,Be,C,Fe,Li,Mg, O,Sn,Y	nrb93#he	1-2, 2-2	*ls*	medium
Li-like	Ar,B,Be,C,Fe,Mg, O,Sn,Y	nrb93#li	1-2, 2-2, 2-3	*ls*	medium
Be-like	Ar,B,C,Fe,Mg, O,Sn,Y	nrb93#be	2-2, 2-3	*ls*	medium
B-like	Ar,C,Fe,Mg, O,Sn,Y	nrb93#b	2-2, 2-3	*ls*	medium
C-like	Ar, Fe,Mg,O,Y	nrb93#c	2-2, 2-3	*ls*	medium
N-like	Ar,Fe,Mg,O,Y	nrb93#n	2-2, 2-3	*ls*	medium
O-like	Ar,Fe,Mg,Y	nrb93#o	2-2, 2-3	*ls*	medium
F-like	Ar,Fe,Mg,Y	nrb93#f	2-2, 2-3	*ls*	medium
Ne-like	Ar,Fe,Mg,Y	nrb93#ne	2-3	*ls*	medium
H-like.	C, N, Ne	mom93#h	1-2	*ls*	medium
He-like	C, N, Ne	mom93#he	1-2, 2-2	*ls*	medium
Li-like	C, N, Ne	mom93#li	1-2, 2-2, 2-3	*ls*	medium
Be-like	C, N, Ne	mom93#be	2-2, 2-3	*ls*	medium
B-like	C, N, O, Ne	mom93#b	2-2, 2-3	*ls*	medium
C-like	N, O, Ne	mom93#c	2-2, 2-3	*ls*	medium
N-like	O, Ne	mom93#n	2-2, 2-3	*ls*	medium
O-like	Ne, Mg	mom93#o	2-2, 2-3	*ls*	medium
F-like	Ne, Mg	mom93#f	2-2, 2-3	*ls*	medium

H-like.	extend. elem. rnge	nrb00#h	1-2	*ls, ic, icr*	high
He-like	extend. elem. rnge	nrbmb00#he	1-2, 2-3	*ls, ic, icr*	high
Li-like	extend. elem. rnge	nrbjc00#li	2-2, 2-3	*ls, ic, icr*	high
Be-like	extend. elem. rnge	nrbjc00#be	2-2, 2-3	*ls, ic, icr*	high
B-like	extend. elem. rnge	nrbza00#b	2-2, 2-3	*ls, ic, icr*	high
C-like	extend. elem. rnge	nrboiz00#c	2-2, 2-3	*ls, ic, icr*	high
N-like	extend. elem. rnge	nrbdmm00#n	2-2, 2-3	*ls, ic, icr*	high
O-like	extend. elem. rnge	nrboiz00#o	2-2, 2-3	*ls, ic, icr*	high
F-like	extend. elem. rnge	oiz00#f	2-2, 2-3	*ls, ic, icr*	high
Ne-like	extend. elem. rnge	nrb00#ne	2-3	*ls, ic*	high
Ne-like	extend. elem. rnge	oiz00#ne	2-3	*ls, ic*	high
Na-like	extend. elem. rnge	za00#na	2-3, 3-3, 3-4	*ls, icm, icr*	high

Notes:

1. The formerly named ‘mom96’ data have been renamed as ‘mom93’ since they are of the same calculation type as for the ‘nrb93’ data. The year number is therefore being used to mark the first year of a calculation type. The new year number ‘00’ has been introduced for the intermediate coupling *ic* dielectronic data. ‘00’ is also used for new *ls* dielectronic data which has extended temperature ranges and is patterned on the *ic* format.
2. Some sequences have been revised with relativistic intermediate coupling calculations (*icr, icm*). Original data sets (*ic*) remain but the directory has been renamed to identify both originators, for example nrb & oiz.

Data lines - ls old format:

Data lines	Format
seq=’SEQSYM’ , *nucchg*=IZ0 , CFORM , ADF09 [CFORM is ’/LS/’ or ’/IC/’ for new form (post 93) data] [CFORM is ’ ’ for old form (93 and earlier) data] [Note keywords are italicised]	1a128: keywords - a2,i,a4,a5
<blank line>	1a128
C24 , *bwnp*= BWNP , *nprnti*= NPRNTI, *nprntf*= NPRNTF [C24 contains ’*parent term’* if *ls* form or ’*parent level’* if *ic* form]	1a128: a24,keywords -f12.1,i,i
<underlines>	1a128
<descriptive headers>	1a128
<underlines>	1a128

do i=1,NPRNTI	
IPA(i),C20, ISPA(i), ILPA(i), C7, C37	i6,5x,1a20,1x,2(i1,1x),a7,1x,a37 [i6,10x,1a20,1x,2(i1,1x),a7,1x,a37 if new form]
[strings c20, C7 and C37 contain CSTRGPA(i), XJPA(i) and WPA(i) respectively]	
enddo	
<blank line>	1a128
C28, *bwnr*= BWNR, C4= NTRM	1a128: keywords -f12.1,i
[C28 contains '*ls resolved term*' if *ls* form or '*ic resolved level*' if *ic* form ; C4 contains '*ntrm*' if *ls* form or '*nlvl*' if *ic* form]	
(prescribed text field)	1a28
(prescribed text field)	1a56
(prescribed text field)	1a56
do indx=1,NTRM	
INDX,INDP,CFGT,IS, IL, XJ, WNRT	i6,5x,1a20,1x,i1,1x,i1,1x,f4.1,1x,f11.1
enddo	
(blank line)	1a80
(prescribed text field),NREP	1a61,i3
(prescribed text field)	1a56
(prescribed text field)(CTRNS(I),I=1, NPRNT*(NPRNT-1)/2)	1a21,10(7x,1a3)
do irep=1,NREP	
IREP,NR,(AAMPM(IN),IN=1,NPRNT*(NPRNT-1)/2)	2i6,9x,10(e10.2)
enddo	
(blank line)	1a80
(blank line)	1a80
(blank line)	1a80
do iprti=1,NPRNT	
(prescribed text field)	1a33
(prescribed text field),IPRTI, TRMPRI, SPNPRI	6x,i2,10x,1a4,9x,i2
(blank line)	1a80
(prescribed text field),(TE(IT),IT=1,MAXT)	11x,10e10.2/11x,10e10.2
do itrm=1,NTRM	
INDX, (ALT(IT),IT=1,MAXT)	i6,5x,10e10.2/11x,10e10.2
enddo	
(blank line)	1a80
(blank line)	1a80
do IPRTF=1,NPRTF	
(prescribed text field)	1a42
(prescribed text field),IPRTF, TRMPRF, SPNPRF,NSYSF	6x,i2,10x,1a4,9x,i2,7x,i2
do isys=1,NSYS	
(prescribed text field)	1a111
(prescribed text field),IS,SPNSYS	1a98,i2,9x,i2
(prescribed text field)	1a7
(prescribed text field)	1a7
do irep=1,NREP	
IREP, (ANT(IT),IT=1,MAXT)	i6,5x,10e10.2/11x,10e10.2
enddo	
enddo	
enddo	
enddo	
(blank line)	1a80
(blank line)	1a80
(prescribed comment line)	1a80
(prescribed comment line)	1a80
(prescribed comment line)	1a80

Variable identification - ls old format:

Name	Meaning	Comment
SEQ	sequence identifier	two chars., blank fill to right
IZ0	nuclear charge	
CCPLG	coupling scheme, '/LS/' =¿ *ls*	
BWNP	binding wave no. of lowest parent (cm-1)	
NPRNTI	no. of initial meta. parents (including grnd parent)	
NPRNTF	no. of final meta. parents (including grnd parent)	
INDP	index of parent	
CFGP	configuration (or Eissner code therefor) for parent.	
ISP	multiplicity of parent (2Sp+1)	
ILP	lotal orbital quantum number (Lp) for parent	
XJP	(statist. weight - 1)/2 of parent term	
WNPI	energy of parent term relative to lowest parent (cm-1)	
BWNR	binding enrgy of lowest term rel. to lowest parent (cm-1)	
NTRM	number of terms in LS-resolved set	
INDX	index value for term	
INDP	index of parent	
CFGT	configuration (or Eissner code thereof) for term.	
IS	multiplicity for level (2*S+1)	
IL	total orbital quantum number for term	
XJ	(statist. weight - 1)/2 for term	
WNRT	energy of term relative to ground (cm-1)	
AATP()	specifies meta. to meta. secondary Auger path for terms	
NREP	number of representative n-shells	
CTRNS()	specifies meta. to meta. secondary Auger path, written as m'-m where m' and m are meta. parent indices.	
IREP	index of representative n-shells	
NR	principal quantum number	
AAMPM()	Auger rate coefficients (sec-1)	
IPRTI	index of initial parent	
TRMPRI	term specification of initial parent	
SPNPRI	spin multiplicity of initial parent	
TE()	prescribed electron temperatures (K)	
MAXT	number of temperatures	
INDX	index of term	
ALT()	diel. coeffts. for term for the initial parent	
IPRTF	index of final parent	
TRMPRF	term specification of finalparent	
SPNPRF	spin multiplicity of final parent	
NSYSF	no. of spin systems assoc. with recomb. to this final parent (1 or 2)	
IS	index of spin system	
SPNSYS	spin multiplicity of spin system	
IREP	index of representative n-shell	
ANT()	dielectronic coefficients for n-shell	

Sample 1: **/home/adas/adas/adf09/nrbjc00#be/jc00_v19ic22.dat**

New format dielectronic data organisation from year 2000 onwards. The data set is in *ic* resolution and refers to parent $n = 2 - 2$ transition type. There is a separate data set for $n = 2 - 3$. The temperature set and temperature dependent data can extend over more that one line - see example 2 for an illustration.

```
SEQ='B '      NUCCHG=23                                                    /IC/
```

```
   PARENT LEVEL INDEXING                BWNP= 138380083.1   NPRNTI= 4   NPRNTF=10
   ---------------------
   INDP         CODE                  S L   WI        WNP
   ----         ----                  - -   --  ----------
     1          2S2                 (1)0( 0.0)        0.0
     2          2S1 2P1             (3)1( 0.0)   298708.6
     3          2S1 2P1             (3)1( 1.0)   318278.2
     4          2S1 2P1             (3)1( 2.0)   367687.0
     5          2S1 2P1             (1)1( 1.0)   630571.5*
     6          2P2                 (3)1( 0.0)   809664.9*
     7          2P2                 (3)1( 1.0)   843929.0*
     8          2P2                 (3)1( 2.0)   877468.8*
     9          2P2                 (1)2( 2.0)   974870.3*
    10          2P2                 (1)0( 0.0)  1171319.4*

   IC RESOLVED LEVEL INDEXING           BWNR= 149287009.6   NLVL=1037
   --------------------------
   INDX  INDP   CODE                  S L   WJ        WNR     AUGER RATES: INDP-INDP' ..
   ----  ----   ----                  - -   --  ----------    -----------
     1     1    2S2 2P1             (2)1( 0.5)        0.0
     2     1    2S2 2P1             (2)1( 1.5)    64359.6
     3     1    2S1 2P2             (4)1( 0.5)   317060.1

. . .

  1033     5    2S1 2P1 8S1         (2)1( 0.5) 10900180.3
  1034     5    2S1 2P1 8S1         (2)1( 1.5) 10900287.2
  1035     6    2P2 7D1             (4)3( 1.5) 10901578.5
  1036     6    2P2 7D1             (4)3( 2.5) 10903276.6
  1037     6    2P2 7F1             (4)4( 2.5) 10906725.0

   NL-SHELL INDEXING & AUGER RATES                    NLREP= 55
   -------------------------------
   ILREP   N   L  M'-M =    2-1       3-1       4-1       3-2       4-2       4-3
   -----   -   -  ----
     1     1   0       0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00
     2     2   0       0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00
     3     2   1       0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00

. . .

    53    10   7       0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00
    54    10   8       0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00
    55    10   9       0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00

   N-SHELL INDEXING & AUGER RATES                     NREP= 42
   ------------------------------
   INREP   N  M'-M =        2-1       3-1       4-1       3-2       4-2       4-3
   -----   -  ----
     1     1          0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00
     2     2          0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00
     3     3          0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00

. . .

    40   700         4.23E+01  2.16E+02  1.71E+02  6.23E+02  4.00E+02  1.54E+03
    41   811         2.03E+01  1.03E+02  8.17E+01  2.99E+02  1.92E+02  7.36E+02
```

```
    42   999          7.15E+00  3.64E+01  2.88E+01  1.05E+02  6.76E+01  2.60E+02


 --------------------------
 PRTI= 1  LVLPRT= (1S  0.0)

   INDX TE=  3.61E+03  7.22E+03        3.61E+05  7.22E+05  1.81E+06  3.61E+06
   ---- ---
     1       4.48E-15  1.98E-13        1.07E-12  5.27E-13  1.67E-13  6.40E-14
     2       1.40E-15  3.43E-14        2.43E-12  1.24E-12  3.99E-13  1.54E-13
     3       1.19E-13  1.71E-13        8.62E-13  4.11E-13  1.26E-13  4.78E-14

. . .

  1035       0.00E+00  6.65E-30        4.09E-16  2.26E-16  7.54E-17  2.93E-17
  1036       0.00E+00  2.57E-29        2.67E-15  1.46E-15  4.83E-16  1.87E-16
  1037       0.00E+00  1.69E-30        3.09E-16  1.74E-16  5.87E-17  2.29E-17

                                           --------------------------
                                           PRTF= 1  LVLPRT= (1S  0.0)
   ILREP
   -----
     3       2.84E-13  1.11E-12        6.36E-12  3.09E-12  9.69E-13  3.69E-13
    31       5.67E-11  4.90E-11        3.37E-13  1.20E-13  3.06E-14  1.08E-14
    32       1.62E-11  2.71E-11        3.52E-13  1.26E-13  3.23E-14  1.15E-14

. . .

    53       0.00E+00  2.19E-30        2.58E-13  1.44E-13  4.82E-14  1.87E-14
    54       0.00E+00  1.99E-30        2.33E-13  1.31E-13  4.37E-14  1.69E-14
    55       0.00E+00  1.68E-30        1.97E-13  1.11E-13  3.69E-14  1.43E-14

   INREP
   -----
     2       2.84E-13  1.11E-12        6.36E-12  3.09E-12  9.69E-13  3.69E-13
     8       1.53E-10  2.29E-10        2.95E-12  1.06E-12  2.71E-13  9.61E-14
     9       2.54E-32  2.26E-21        2.30E-12  1.07E-12  3.19E-13  1.19E-13

. . .

    40       0.00E+00  4.40E-42        4.55E-16  5.46E-16  2.89E-16  1.31E-16
    41       0.00E+00  3.00E-42        2.94E-16  3.52E-16  1.87E-16  8.45E-17
    42       0.00E+00  1.71E-42        1.58E-16  1.89E-16  1.00E-16  4.53E-17

                                           --------------------------
                                           PRTF= 2  LVLPRT= (3P  0.0)
   ILREP
   -----
     3       2.81E-10  2.44E-10        6.39E-12  3.01E-12  9.39E-13  3.59E-13
     4       2.17E-15  1.08E-13        1.60E-13  7.35E-14  2.21E-14  8.28E-15
     5       3.99E-15  2.24E-13        4.27E-13  1.96E-13  5.91E-14  2.22E-14

. . .

    53       2.83E-23  2.62E-21        4.83E-21  2.10E-21  6.12E-22  2.27E-22
    54       1.28E-26  1.18E-24        2.22E-24  9.76E-25  2.87E-25  1.07E-25
```

```
    55       1.41E-30  1.31E-28        1.51E-27  1.38E-27  6.35E-28  2.75E-28

   INREP
   -----
     2       2.81E-10  2.44E-10        6.39E-12  3.01E-12  9.39E-13  3.59E-13
     3       8.29E-15  5.13E-13        1.04E-12  4.78E-13  1.44E-13  5.39E-14
     4       4.79E-15  2.96E-13        6.07E-13  2.79E-13  8.39E-14  3.15E-14

. . .

    40       0.00E+00  2.81E-45        4.51E-20  8.75E-20  6.16E-20  3.06E-20
    41       0.00E+00  1.01E-45        2.90E-20  5.63E-20  3.96E-20  1.97E-20
    42       0.00E+00  0.00E+00        1.54E-20  3.01E-20  2.12E-20  1.05E-20

                                        --------------------------
                                        PRTF= 3  LVLPRT= (3P  1.0)

. . .

                                       --------------------------
                                        PRTF=10  LVLPRT= (1S  0.0)
. . .

 --------------------------
 PRTI= 2  LVLPRT= (3P  0.0)

. . .

 --------------------------
 PRTI= 4  LVLPRT= (3P  2.0)

. . .

    T(K)     ALFT( 1)  ALFT( 2)        ALFT( 7)  ALFT( 8)  ALFT( 9)  ALFT(10)
    ----     --------  --------        --------  --------  --------  --------
  3.61E+03   2.59E-09  2.35E-10
  7.22E+03   3.02E-09  3.10E-10
  1.81E+04   1.73E-09  2.34E-10

. . .

C---------------------------------       ---------------------------------------
C
C---------------------------------       ---------------------------------------
```

Sample 2: **/home/adas/adas/adf09/nrbjc00#be/jc00_v19ls22.dat**

The *ls* form of the new format is illustrated below. Note the key phrases appropriate for term coupling - otherwise layout is as for the *ic* example 1 above. There are no separate high representative *nl*-shell blocks for the *ls* case.

```
SEQ='B '     NUCCHG=23                                                  /LS/

   PARENT TERM INDEXING                BWNP= 137430030.2   NPRNTI= 2   NPRNTF= 6
   --------------------
   INDP         CODE                S L   WI        WNP
```

```
   ----         ----                - -   --  ----------
     1          2S2                (1)0( 0.0)        0.0
     2          2S1 2P1            (3)1( 4.0)   296736.7
     3          2S1 2P1            (1)1( 1.0)   579190.7*
     4          2P2                (3)1( 4.0)   765813.2*
     5          2P2                (1)2( 2.0)   861316.0*
     6          2P2                (1)0( 0.0)  1064133.0*

   LS RESOLVED TERM INDEXING           BWNR= 148261739.7   NTRM= 396
   -------------------------
   INDX  INDP   CODE                S L   WJ        WNR       AUGER RATES:  INDP-INDP' ..
   ----  ----   ----                - -   --  ----------      -----------
     1     1    2S2 2P1            (2)1( 2.5)        0.0
     2     2    2S1 2P2            (4)1( 5.5)   269317.6
     3     2    2S1 2P2            (2)2( 4.5)   502352.5

. . .
   394     6    2P2 6F1            (2)3( 6.5) 10794336.1
   395     6    2P2 6G1            (2)4( 8.5) 10795368.6
   396     6    2P2 6H1            (2)5(10.5) 10795422.4

   N-SHELL INDEXING & AUGER RATES                      NREP= 42
   ------------------------------
   IREP    N  M'-M =        2-1
   -----   -  ----
     1     1           0.00E+00
     2     2           0.00E+00
     3     3           0.00E+00

. . .
    40   700           1.94E+02
    41   811           9.27E+01
    42   999           3.27E+01



 --------------------------------
 PRTI= 1  TRMPRT= (1S)  SPNPRT= 1

   INDX TE=  3.61E+03  7.22E+03      1.81E+05  3.61E+05  7.22E+05  1.81E+06  3.61E+06
   ---- ---  7.22E+06  1.81E+07      3.61E+08  7.22E+08  1.81E+09  3.61E+09
     1       2.56E-14  1.32E-12      5.56E-12  3.27E-12  1.58E-12  4.93E-13  1.88E-13
             6.91E-14  1.79E-14      2.03E-16  7.18E-17  1.82E-17  6.43E-18
     3       4.64E-14  3.14E-12      1.89E-11  1.08E-11  5.14E-12  1.59E-12  6.06E-13
             2.23E-13  5.76E-14      6.53E-16  2.31E-16  5.85E-17  2.07E-17

. .
.
   394       0.00E+00  2.38E-34      5.21E-15  5.68E-15  3.67E-15  1.36E-15  5.50E-16
             2.08E-16  5.48E-17      6.28E-19  2.22E-19  5.63E-20  1.99E-20
   395       0.00E+00  1.71E-35      7.60E-16  8.60E-16  5.69E-16  2.14E-16  8.70E-17
             3.30E-17  8.69E-18      9.98E-20  3.53E-20  8.94E-21  3.16E-21
   396       0.00E+00  1.26E-37      6.44E-18  7.35E-18  4.89E-18  1.85E-18  7.50E-19
             2.84E-19  7.50E-20      8.62E-22  3.05E-22  7.72E-23  2.73E-23


 ----------------------------------------
```

```
 PRTF= 1  TRMPRT= (1S)  SPNPRT= 1  NSYS= 1
                                                                                 -----------------
                                                                                 SYS= 1  SPNSYS= 2
   IREP
   ----
     2       2.56E-14  1.32E-12      5.56E-12  3.27E-12  1.58E-12  4.93E-13  1.88E-13
             6.91E-14  1.79E-14      2.03E-16  7.18E-17  1.82E-17  6.43E-18
     9       9.91E-24  4.10E-17      4.10E-12  2.04E-12  8.58E-13  2.41E-13  8.81E-14
             3.17E-14  8.10E-15      9.11E-17  3.22E-17  8.15E-18  2.88E-18
    10       4.33E-40  3.68E-25      2.25E-12  1.63E-12  8.24E-13  2.58E-13  9.81E-14
             3.60E-14  9.29E-15      1.05E-16  3.72E-17  9.42E-18  3.33E-18

. . .

    40       0.00E+00  0.00E+00      1.48E-16  5.19E-16  5.78E-16  2.91E-16  1.29E-16
             5.14E-17  1.39E-17      1.63E-19  5.75E-20  1.46E-20  5.15E-21
    41       0.00E+00  0.00E+00      9.51E-17  3.34E-16  3.72E-16  1.87E-16  8.34E-17
             3.31E-17  8.96E-18      1.05E-19  3.70E-20  9.38E-21  3.32E-21
    42       0.00E+00  0.00E+00      5.09E-17  1.79E-16  1.99E-16  1.00E-16  4.47E-17
             1.77E-17  4.80E-18      5.60E-20  1.98E-20  5.02E-21  1.78E-21



. . .



    T(K)     ALFT( 1)  ALFT( 2)      ALFT( 6)  ALFT( 7)  ALFT( 8)  ALFT( 9)  ALFT(10)
    ----     --------  --------      --------  --------  --------  --------  --------
  3.61E+03   1.51E-13  1.00E-16
  7.22E+03   9.36E-12  7.29E-14
  1.81E+04   7.00E-11  3.70E-12

. . .

  7.22E+08   4.75E-15  5.07E-16
  1.81E+09   1.20E-15  1.28E-16
  3.61E+09   4.25E-16  4.53E-17

C-----------------------------------------------------------------------------------
C
C
C
C-----------------------------------------------------------------------------------
```

Sample 3: **/home/adas/adas/adf09/nrb93#li/nrb93#li_sn47ls12.dat**

Old format dielectronic data organisation before year 2000. The data set is in *ls* resolution and refers to parent $n = 2 - 2$ transition type. Note that there are no separate high representative *nl*-shell blocks and no data block at the end for the total dielectronic rate coefficient in the old format.

```
SEQ='BE'     NUCCHG=50

   PARENT TERM INDEXING            BWNP= 606126998.2   NPRNTI= 2   NPRNTF= 2
   --------------------
```

```
   INDP    CODE                 S L   WI        WNP
   ----    ----                 - -   --  ----------
     1     1S2 2S1             (2)0( 0.5)        0.0
     2     1S2 2P1             (2)1( 2.5)   750575.7

   LS RESOLVED TERM INDEXING       BWNR= 668924621.0    NTRM=166
   -------------------------
   INDX    CODE                 S L   WJ        WNR
   ----    ----                 - -   --  ----------
     1     1S2 2S2             (1)0( 0.0)        0.0
     2     1S2 2S1 2P1         (3)1( 4.0)   683850.3
     3     1S2 2S1 2P1         (1)1( 1.0)  1315281.3

. . .
   164     1S2 2P1 8F1         (1)2( 2.0) 59761223.5
   165     1S2 2P1 8D1         (1)1( 1.0) 59761725.3
   166     1S2 2P1 8D1         (1)3( 3.0) 59761798.8

   N-SHELL INDEXING & AUGER RATES                      NREP= 42
   ------------------------------
   IREP    N  M'-M =        2-1
   ----    -  ----
     1     1           0.00E+00
     2     2           0.00E+00
     3     3           0.00E+00

. . .
    40   700           0.00E+00
    41   811           0.00E+00
    42   999           0.00E+00



 --------------------------------
 PRTI= 1  TRMPRT= (2S)  SPNPRT= 2

   INDX TE=  2.21E+06  4.42E+06  1.10E+07  2.21E+07       4.42E+08  1.10E+09  2.21E+09
   ---- ---
     1       0.00E+00  1.55E-33  1.98E-22  5.60E-19       3.46E-17  1.16E-17  4.48E-18
     2       0.00E+00  7.91E-31  1.11E-19  3.30E-16       2.60E-14  8.84E-15  3.45E-15
     3       0.00E+00  2.20E-31  3.37E-20  1.03E-16       7.81E-15  2.64E-15  1.03E-15

. . .

   164       0.00E+00  1.99E-44  2.52E-28  3.29E-23       7.17E-20  2.67E-20  1.07E-20
   165       0.00E+00  2.23E-43  2.82E-27  3.69E-22       8.03E-19  2.98E-19  1.20E-19
   166       0.00E+00  5.16E-43  6.54E-27  8.54E-22       1.86E-18  6.92E-19  2.78E-19


 ----------------------------------------
 PRTF= 1  TRMPRT= (2S)  SPNPRT= 2  NSYS= 2
                                                        -----------------
                                                        SYS= 1  SPNSYS= 1
   IREP
   ----
     2       0.00E+00  2.21E-31  3.39E-20  1.04E-16       7.84E-15  2.65E-15  1.03E-15
     3       0.00E+00  9.33E-37  1.36E-22  4.01E-18       2.12E-15  7.53E-16  2.98E-16
```

```
     4       0.00E+00  1.16E-38  1.76E-23  1.13E-18       1.26E-15  4.57E-16  1.83E-16

. . .

    39       0.00E+00  4.16E-47  1.25E-30  2.17E-25       6.20E-22  2.32E-22  9.36E-23
    40       0.00E+00  1.86E-47  5.58E-31  9.69E-26       2.77E-22  1.04E-22  4.18E-23
    41       0.00E+00  1.19E-47  3.59E-31  6.23E-26       1.78E-22  6.67E-23  2.69E-23
    42       0.00E+00  6.39E-48  1.92E-31  3.33E-26       9.52E-23  3.57E-23  1.44E-23

                                                          -----------------
                                                          SYS= 2  SPNSYS= 3
   IREP
   ----
     2       0.00E+00  7.91E-31  1.11E-19  3.30E-16       2.60E-14  8.84E-15  3.45E-15
     3       0.00E+00  2.88E-36  4.17E-22  1.22E-17       6.45E-15  2.29E-15  9.07E-16
     4       0.00E+00  2.36E-38  3.59E-23  2.30E-18       2.56E-15  9.30E-16  3.71E-16

. . .

    40       0.00E+00  4.53E-47  1.32E-30  2.27E-25       6.43E-22  2.41E-22  9.71E-23
    41       0.00E+00  2.92E-47  8.49E-31  1.46E-25       4.14E-22  1.55E-22  6.25E-23
    42       0.00E+00  1.56E-47  4.54E-31  7.82E-26       2.21E-22  8.29E-23  3.34E-23


. . .


C-------------------------------------------------------------------------------
C ADAS ADF09 DATA - SCCS info: @(#)$Header:  ....
C-------------------------------------------------------------------------------
```

A.7 *adf11*: iso-nuclear master files

A.8 *adf15*: photon emissivity coefficients

A.9 *adf23*: state selective electron impact ionisation coefficients

The data sets provide tabular state selective direct ionisation rate coefficients between initial ionising and final ionised states at *ca*, *ls* or *ic* resolution. The tabular data also may include electron impact excitation rate coefficients to autoionising states and Auger yields allowing detailed evaluation of excitation-autoionisation channels.

Data mnemonic: cio
Data root: **/home/adas/adas/adf23/**
Last update: 27 November 2008
Utilising subroutines: **xxdata_23.for**

Formatted files to adf23 specification:

Iso-elec. Library	Members	Resol.	Comments	Quality
grf95#he	th_b3ls - th_o6ls	ls	Griffin	High
grf95#he	exp_b3ls - exp_o6ls	ls	Griffin	High
grf95#li	th_b2ls - th_o5ls	ls	Griffin	High
grf95#be	th_b1ls - th_o4ls	ls	Griffin	High
grf95#b	th_b0ls - th_o3ls	ls	Griffin	High
grf95#c	th_c0ls - th_o2ls	ls	Griffin	High
grf95#n	th_n0ls - th_o1ls	ls	Griffin	High
grf95#n	exp_n0ls - exp_o1ls	ls	Griffin	High
grf95#o	th_o0ls, exp_o0ls	ls	Griffin	High
grf97#cl	ar1	ls	Griffin	High
sdl07#h	ca#h0 - ca#w73	ca	Loch - seq.subset	medium
sdl07#h	ic#h0 - ic#w73	ic	Loch - seq.subset	High
sdl07#he	ca#he0 - ca#w72	ca	Loch - seq.subset	medium
sdl07#he	ic#he0 - ic#w72	ic	Loch - seq.subset	High
sdl07#li	ca#li0 - ca#w71	ca	Loch - seq.subset	medium
sdl07#li	ic#li0 - ic#w71	ic	Loch - seq.subset	High

Iso-nuc. Library	Members	Resol.	Comments	Quality
sdl08#18	ca#ar0 - ca#ar17	ca	Loch	medium
sdl08#50	ca#sn0 - ca#sn49	ca	Loch	medium
sdl08#54	ca#xe0 - ca#xe53	ca	Loch	medium
sdl08#74	ca#w0 - ca#w73	ca	Loch	medium

Notes:

Data lines:

Data lines	Format
seq ='SEQ' , nucchg =IZ0 , type ='CTYPE' , ADF23	1a80: keywords - a2,i,a2,a5
<blank line>	1a80
c10 , bwnf = BWNF nprf = NLVL_F	1a80: a10,keywords -f12,i3
[c10 has form 'aa+nn aaaa' where aa = elem. symb., nn= final ion charge and aaaa = conf/term/levl if CTYPE=ca/ls/ic] [characters are left justified and numbers right justified]	
<underlines>	1a80
<descriptive headers>	1a80
<underlines>	1a80
for i=1,NLVL_F	
IA_F(i) , CODE_F(i) , CSTRGA_F(i) , cnn , WA_F(i)	i5,a1,a,a,f13
[cnn has the form '(ISA_F(i))ILA_F(i)(XJA_F(i))' if CTYPE=ls/ic] and cnn has the form '(XJA_F(i))' if CTYPE=ca]	
endfor	
<blank line>	1a80
c10 , bwni = BWNI c4 = NLVL_I	1a80: a10,keywords -f12,i3
[c10 has form 'aa+nn aaaa' where aa = elem. symb., nn= initial ion charge and aaaa = conf/term/levl if CTYPE=ca/ls/ic]	
[c4 has form 'aaaa' where aaaa = ncfg/ntrm/nlvl if CTYPE=ca/ls/ic]	
<underlines>	1a80
<descriptive headers>	1a80
<underlines>	1a80
do i=1,NLVL_I	
IA_I(i) , CODE_I(i) , CSTRGA_I(i) , cnn , WA_I(i)	i5,a1,a,a,f13
[cnn has the form '(ISA_I(i))ILA_I(i)(XJA_I(i))' if CTYPE=ls/ic and cnn has the form '(XJA_I(i))' if CTYPE=ca]	
enddo	
<blank line>	1a80
<underlines>	1a80
do i=1,NMETI	
<underlines>	1a80
<descriptive headers>	1a80
<underlines>	1a80
meti*= IMET(i)	1a80: keywords - i
te= (TEA_ION(i,it),it=1,NTE)	1a256:keyword - d9.2,21d10.2
<underlines>	1a80
for j = 1 , NLVN_F	
INDF , (QRED_ION(i,indf,it),it=1,NTE)	i5,18x,22d10.2
endfor	
<blank line>	1a80
c20	a20
[c20 has form ' aa+nn Auger yields' where aa = elem. symb. and nn= final ion charge]	
<underlines>	1a80

```
      c25 , (INDF_A(i,k),k=1,NF_A(i))                1a25,i8,21i10
          [c25 has the form ' indi /indf' ]
      <underlines>                                   1a80
      do j = 1 , NLVL_i
          INDI , (YLD_A(i,indi,k),k=1,NF_A(i))       i5,18x,22d10.2
      enddo
      <blank line>                                   1a80
      <descriptive headers>                          1a80
      <underlines>                                   1a80
      te= (TEA_EXC(i,it),it=1,NTE)                   1a256:keyword - d9.2,21d10.2
      <underlines>                                   1a80
      do j = 1 , NLVL_i
          INDI , (QRED_EXC(i,indi,it),it=1,NTE)      i5,18x,22d10.2
      enddo
      <underlines>                                   1a80
enddo
```

Variable identification:

Name	Meaning	Comment
SEQ	iso-electronic sequence	two characters, blank fill to right
IZ0	nuclear charge	
CTYPE	ca, ls or ic denoting resolution	two characters
BWNF	ionisation potential of final state ion (cm-1)	
NLVL_F	no. of final ion configs, terms or levels	depending on resolution
IA_F()	index of final states	
CODE_F()	code for metastables of final state ion	* indicates a metastable
CSTRGA_F()	configuration string of final states	standard notation
ISA_F()	multiplicity	only for ls and ic resolution
ILA_F()	total orbital quantum number	only for ls and ic resolution
XJA_F()	(stat. wt.-1)/2 or J	J only for ic resolution
BWNI	ionisation potential of initial state ion (cm-1)	
NLVL_I	no. of initial ion configs, terms or levels	depending on resolution
IA_I()	index of initial states	
CODE_I()	code for metastables of initial state ion	* indicates a metastable
CSTRGA_I()	configuration string of initial states	standard notation
ISA_I()	multiplicity	only for ls and ic resolution
ILA_I()	total orbital quantum number	only for ls and ic resolution
XJA_I()	(stat. wt.-1)/2 or J	J only for ic resolution
NMETI	no. of metastables of initial ion	
IMETI()	pointers to metastables in initial state index	
TEA_ION(,)	elec. temperatures (K) of ionis. coefft table	1st dim: initial metastable index, 2nd dim: te index
NTE	number of elec. temperatures	
INDF	current final state metastable index	
QRED_ION(,,)	scaled ionis. coeffts (cm3 s-1) table	1st dim: initial metastable index, 2nd dim: final state index 3rd dim: te index
NF_A()	no. of final states for Auger yield table	1st dim: initial metastable index,
INDF_A(,)	final state index for Auger yield table	1st dim: initial metastable index, 2nd dim: final state index
YLD_A(,,)	Auger yield table	1st dim: initial metastable index, 2nd dim: initial state index 3rd dim: final state index
TEA_EXC(,)	elec. temperatures (K) of excit. coefft table	1st dim: initial metastable index, 2nd dim: te index
INDI	current initial state index	
QRED_EXC(,,)	scaled excit. coeffts (cm3 s-1) table	1st dim: initial metastable index, 2nd dim: initial excited state index 3rd dim: te index

Sample 1: **/home/adas/adas/adf23/adf23_sample_ca_without-nrep.dat**

```
seq ='sn'    nucchg = 50    type = 'ca'                                        ADF23

  sn+ 1 conf indexing        bwnf =    116138.9     nprf =  3
  -------------------
  indf    code                            WJ       wnf
  ----    ----                            --  ------------
    1*  4dA 5s2 5p1               (     2.5)          0.0
    2   4dA 5s1 5p2               (    14.5)      55559.2
    3   4s2 4p6 4d9 5s2 5p2       (    74.5)     204542.3

  sn+ 0 conf indexing        bwni =     57118.8     ncfg =  5
  -------------------
  indi    code                            WJ       wni
  ----    ----                            --  ------------
    1*  4dA 5s2 5p2               (     7.0)          0.0
    2   4p5 4dA 4f1 5s2 5p2       (   629.5)     848246.4
    3   4s2 4p5 4dA 5s2 5p3       (    59.5)     788239.3
    4   4p5 4dA 5s2 5p2 5d1       (   449.5)     840261.6
    5   4p5 4dA 5s2 5p2 5f1       (   629.5)     850666.1

------------------------------------------------------------------------------
meti*= 1

  ionis rates
  -----------
  indf               Te= 2.00D+03  5.00D+03  1.00D+04  2.00D+04  5.00D+04  1.00D+05
  ----               ---
    1                    6.13D-08  1.11D-07  1.54D-07  2.02D-07  2.73D-07  3.23D-07
    2                    6.45D-09  1.68D-08  2.59D-08  3.68D-08  5.52D-08  7.08D-08
    3                    2.53D-10  2.04D-09  4.40D-09  7.51D-09  1.55D-08  3.13D-08

  sn+ 1 Auger yields
  ------------------
  indi              \indf       1         3
  ----              -----
    2                    1.00D+00  5.00D+00
    3                    1.00D+00  2.00D+00
    4                    1.00D+00  3.00D+00
    5                    1.00D+00  1.00D+00

  excit rates
  -----------
  indi               Te= 2.00D+03  5.00D+03  1.00D+04  2.00D+04  5.00D+04  1.00D+05
  ----               ---
    2                    1.00D-08  1.00D-08  6.91D-05  6.93D-05  7.00D-05  7.11D-05
    3                    1.00D-08  2.74D+02  4.08D-01  4.82D-01  6.79D-01  9.44D-01
    4                    1.00D-08  1.00D-08  4.32D-02  4.67D-02  5.53D-02  6.51D-02
    5                    1.00D-08  1.00D-08  6.09D-05  6.12D-05  6.19D-05  6.31D-05

C------------------------------------------------------------------------------
C
C  Sample:  ionising ion sn+0
C           configuration average
C           single metastable (ground) state for final ion sn+1
```

```
C           single metastable (ground) state for initial ion sn+0
C           includes direct ionisation
C           includes excitation to a set of low auto-ionising configurations
C           no higher bundle-n shells
C
C           * is used to identify initial and final metastables. The indexing
C           should avoid ambiguity even if * is omitted.
C
C           two or more electron shake-off would require further final ion
C           blocks in addition to the adjacent parent ion.
C           shake-off after direct ionisation would require further final ion
C           blocks in addition to the adjacent parent ion.
C
C-------------------------------------------------------------------------------
```

A.10 *adf32*: drivers for ADAS802 ionisation calculations

The format provides configuration and other control parameters for distorted wave electron impact collisional ionisation rate coefficient calculations. They have no use outside ADAS802 calculations. The format is a sequential link of a number of sub-formats driving the various steps of the calculation. The sub-formats are *.seq.*, *.indx*, *.cfg*, *_ionsn_dirct_<i><j>.dat*, *_ionsn_indirct_<i><j>.dat*, *_autos_in*, *_di_rcg_1.dat* and *_di_rcg_2.dat*. They are either auto-generated in background execution of the ionisation code (see ADAS8#2) or are created from interactive data input on-line (see ADAS802). There is no necessity for their archive storage for the general ADAS user. The format is systematically archived in central ADAS heavy element calculations for completeness of production records.

Data mnemonic:
Data root: **/home/adas/adas/adf32/**
Last update: Jan 15, 2009
Utilising subroutines: **ADAS802**. Note no *read_adf32* is provided in ADAS.

Formatted files to adf32 specification:

Element	Directory	Form	Members
Ar	argon	composite	ar0.dat - ar17.dat
Kr	krypton	composite	kr0.dat - kr35.dat
Xe	xenon	composite	xe0.dat - xe53.dat
Ag	silver	composite	ag0.dat - ag46.dat
Sn	tin	composite	sn0.dat - sn49.dat
W	tungsten	composite	w0.dat - w73.dat

A.11 *adf34*: drivers for ADAS801 structure calculations

The format provides configuration and other control parameters for Cowan atomic structure calculations. They have no use outside Cowan calculations. There are four variants called the *primary*, *_inst*, *_ls_pp*, *_ic_pp* datasets. The latter three are relevant only to the intermediate steps of automatic Cowan calculations. They are either auto-generated in background execution of the Cowan code (see ADAS8#1) or are created from interactive data input on-line (see ADAS801). There is no necessity for their archive storage for the general ADAS user and they are included here only for completeness. The *primary* forms the main input dataset to ADAS801. It is of value to archive as a record (and for repeatability) of a structure calculation. It can also be prepared by hand or simply modified (for example by including an extra configuration). The format is systematically archived in central ADAS heavy element calculations.

Data mnemonic:
Data root: **/home/adas/adas/adf34/**
Last update: Dec 18, 2008
Utilising subroutines: **ADAS801**. Note no *read_adf34* is provided in ADAS.

Formatted files to adf34 specification:

Element	Directory	Form	Members
Ar	argon	primary	ar0.dat - ar17.dat
		_inst	ar0_inst.dat - ar17_inst.dat
		_ls_pp	ar0_ls_pp.dat - ar17_ls_pp.dat
		_ic_pp	ar0_ls_pp.dat - ar17_ls_pp.dat
Kr	krypton	primary	kr0.dat - kr35.dat
Xe	xenon	primary	xe0.dat - xe53.dat
Ag	silver	primary	ag0.dat - ag46.dat
Sn	tin	primary	sn0.dat - sn49.dat
W	tungsten	primary	w0.dat - w73.dat

Data lines (primary variant):

Data lines	Format
C80	1a80
do i=1,NCONFG	
IZ0, IZ1, CTITLA(i), CFCOM(i)	2i5,1a18,3x,1a20
enddo	
-1	i5

Variable identification (primary variant):

Name	Meaning	Comment
C80	control string - preferred settings given should be retained [Refer to Cowan text - 'Atomic Structure' for detail of control settings. ADAS801 uses a default set]	
IZ0	nuclear charge	
IZ1	effective ion charge [Usually taken as ion charge+1, this can give an incorrect total electron count for complex ions. The Cowan prescription is based on reference to the nearest rare gas core (and may be negative). ADAS automatic heavy species codes have a robust IZ1 specifier. The user varying this must check the electron count from the *rcn* step.]	
CTITLA()	string for user configuration naming convenience in output - preferred shown	
CFCOM()	configuration specifications in Cowan format	

Sample (primary variant): **/home/adas/adas/adf34/argon/ar3.dat**

```
2  -5    2   10  1.0    5.d-09    5.d-11-2  0130    1.0 0.65  0.0  0.5
   18    4   Ar ground z1= 3 1   3s2  3p3
   18    4   Ar cfg 01       1   3s2  3p2  4p1
   18    4   Ar cfg 02       1   3s2  3p2  4f1
   18    4   Ar cfg 03       0   3s2  3p2  3d1
   18    4   Ar cfg 04       0   3s2  3p2  4s1
   18    4   Ar cfg 05       0   3s2  3p2  4d1
   18    4   Ar cfg 06       0   3s1  3p4
   -1
```

Sample (_inst variant): **/home/adas/adas/adf34/argon/ar3_inst.dat**

```
z0 18
zi  3
parity-1  3
parity-2  4
E2 3
M1 3
scale 75 95 75 75 75
temperature 25
 2.00e+02 3.00e+02 5.00e+02 7.00e+02 1.00e+03 1.50e+03 2.00e+03 3.00e+03
 5.00e+03 7.00e+03 1.00e+04 1.50e+04 2.00e+04 3.00e+04 5.00e+04 7.00e+04
 1.00e+05 1.50e+05 2.00e+05 3.00e+05 5.00e+05 1.00e+06 2.00e+06 5.00e+06
 1.00e+07
```

Sample (_ls_pp variant): **/home/adas/adas/adf34/argon/ar3_ls_pp.dat**

```
1
Hugh Summers
10/08/07
5
C
C  Cowan plane wave Born method
C
C  Scale factors 75 95 75 75 75
C
 &FILES ifgfile = './ifg#adf34_argon_ar3.dat' , outfile = 'adf04_copmm#18_ls#ar3.dat' &END
 &OPTIONS  ip =        482782.27, coupling = 'LS' ,  aval = 'YES' ,  isonuclear = 'YES',
           quantity = 'RATES', lweight = 'NO' ,  comments = 2,  numtemp = 14 , &END
  1  2  3  5  7  9 11 12 13 14 15 17 19 20
```

Sample (_ic_pp variant): **/home/adas/adas/adf34/argon/ar3_ic_pp.dat**

```
1
Hugh Summers
10/08/07
5
C
C  Cowan plane wave Born method
C
C  Scale factors 75 95 75 75 75
C
```

```
&FILES ifgfile = './ifg#adf34_argon_ar3.dat' , outfile = 'adf04_copmm#18_ic#ar3.dat' &END
&OPTIONS  ip =        482782.27, coupling = 'IC' ,  aval = 'YES' ,  isonuclear = 'YES',
          quantity = 'RATES', lweight = 'NO' ,  comments = 2,  numtemp = 14 , &END
 1  2  3  5  7  9 11 12 13 14 15 17 19 20
```

A.12 *adf40*: envelope feature photon emissivity coefficients

The format provides a pixellated tabulation of one or more spectral intervals containing the line broadened emissivities of the spectrum of an ion of an element. Analogous to the emissivity coefficients of *adf15*, the feature emissivity coefficients are collisional-radiative quantities, tabulated for ranges of electron temperatures and electron densities. The spectrum line content of a feature emissivity coefficient depends on the data input (normally an adf04 data set) to the collisional-radiative calculation. The types *exc*, *rec* and *cx* types and resolutions *ca*, *ls* and *ic*, used for *adf15* also apply to *adf40*. Since the wavelength intervals of feature emissivity coefficients are usually set up for particular local spectrometers, the central ADAS database content of this format is restricted to a few generic ranges, such as VUV, XUV and visible.

Data mnemonic: fpec
Data root: **/home/adas/adas/adf40/**
Last update: Dec 10, 2008
Utilising subroutines: **read_adf40.pro**, **xxdata_40.pro**

Formatted files to adf54 specification:

Element	Directory	Sub-dir. layer	Resolution	Members
Xe	copmm#54	xe_01	ca	ca#xe0.dat - ca#xe53.dat
			ls	ls#xe0.dat - ls#xe53.dat
			ic	ic#xe0.dat - ic#xe53.dat
Sn	copmm#50	xe_01	ca	ca#sn0.dat - ca#sn49.dat
			ls	ls#sn0.dat - ls#sn49.dat
			ic	ic#sn0.dat - ic#sn49.dat

Data lines:

Data lines	Format
NSEL, SYM, IZ, TEXT,RCODE	i5,4x,'/',1a3,i2,1a54,'/',1a2,'/'
do isel= 1 to NSEL	
FCODE, NPIX , NDENS , NTE , FILMEM, TYPE , INDM , ISEL	a6,i6,2i4,2c8,i2,i5
[NB. '/' and 'code=' delimited]	
WVMIN, WVMAX	2f12.5
(DENS(in), in=1,NDENS)	8e9.2
(TE(it), it=1,NTE)	8e9.2
do in = 1 to NDENS	
do it = 1 to NTE	
(FPEC(ipix,in,it,ISEL), ipix=1,NPIX)	8e9.2
enddo	
enddo	
enddo	

Variable identification:

Name	Meaning	Comment
NSEL	number of transitions available	
SYM	element symbol in form	
IZ	charge of the ion	
TEXT	information	
RCODE	resolution code; LS=¿ ls-resolution; IC=¿ intermediate coupling	
FCODE	Filter character code if present	
NPIX	Number of pixels	
NDENS	number of densities	
NTE	number of temperatures	
FILMEM	source specific ion excitation file	
TYPE	type of photon emissivity (excit, recomb, cx)	
INDM	associated metastable index in source file	
ISEL	transition index	
WVMIN	minimum wavelength of spectral interval (Angstrom)	
WVMAX	maximum wavelength of spectral interval (Angstrom)	
DENS()	electron densities (cm-3)	
TE()	electron temperatures (eV)	
FPEC(,,)	finite density feature photon emissivity coefficients (cm3 s-1)	
	1st parameter pixel index	
	2nd parameter electron density index	
	3rd parameter electron temperature index	

Sample: **/home/adas/adas/adf40/xe_10.dat**

```
    2    /Xe+10 envelope feature photon emissivity coefficients      /IC/
ft1235   128  24  24 /filmem =         /type = f_excit  /indm = 1/isel =    1
   200.00000  1000.00000
 1.00e+01 1.00e+02 1.00e+03 1.00e+04 1.00e+05 1.00e+06 3.00e+06 1.00e+07
 3.00e+07 1.00e+08 3.00e+08 1.00e+09 3.00e+09 1.00e+10 3.00e+10 1.00e+11
 3.00e+11 1.00e+12 3.00e+12 1.00e+13 3.00e+13 1.00e+14 3.00e+14 1.00e+15
 4.31e-02 6.03e-02 8.62e-02 1.29e-01 1.72e-01 2.59e-01 4.31e-01 6.03e-01
 8.62e-01 1.29e+00 1.72e+00 2.59e+00 4.31e+00 6.03e+00 8.62e+00 1.29e+01
 1.72e+01 2.59e+01 4.31e+01 6.03e+01 8.62e+01 1.29e+02 1.72e+02 2.59e+02
 1.00e-74 1.00e-74 1.00e-74 1.00e-74 1.00e-74 1.00e-74 2.82e-30 3.22e-24
  . . .
 9.19e-10 1.56e-09 2.51e-09 3.14e-09 3.76e-09 4.36e-09 4.71e-09 5.03e-09
ft1235   128  24  24 /filmem =         /type = f_excit  /indm = 1/isel =    2
    10.00000   100.00000
 1.00e+01 1.00e+02 1.00e+03 1.00e+04 1.00e+05 1.00e+06 3.00e+06 1.00e+07
 3.00e+07 1.00e+08 3.00e+08 1.00e+09 3.00e+09 1.00e+10 3.00e+10 1.00e+11
 3.00e+11 1.00e+12 3.00e+12 1.00e+13 3.00e+13 1.00e+14 3.00e+14 1.00e+15
 4.31e-02 6.03e-02 8.62e-02 1.29e-01 1.72e-01 2.59e-01 4.31e-01 6.03e-01
 8.62e-01 1.29e+00 1.72e+00 2.59e+00 4.31e+00 6.03e+00 8.62e+00 1.29e+01
 1.72e+01 2.59e+01 4.31e+01 6.03e+01 8.62e+01 1.29e+02 1.72e+02 2.59e+02
 7.95e-14 5.90e-14 5.27e-14 7.86e-14 1.12e-13 1.56e-13 1.68e-13 1.47e-13
  . . .
 1.28e-14 8.71e-15 6.27e-15 3.54e-15 2.15e-15 2.51e-15 4.70e-15 1.01e-14
C-----------------------------------------------------------------------
c
c  envelope feature photon emissivity coefficients:
c
c  information
c  -----------
c
c  nuclear charge = 54
c  ion charge + 1 = 11
c
c  specific ion file :  /home/adas/adas/adf04/copmm#54/ls#xe10.dat
c  expansion file    :  no projection data was used in this case
c
c  no ionisation data has been included
c
c  options  : lnorm=T  lpsel=F  lzsel=F  lionsel=T
c             lhsel=F  lrsel=F  lisel=F  lnsel  =F
c
c  population processing code:  adas810
c
c
c
c  isel  iwvrg   wavelength range (ang)   type   metastable  imet nmet  ip
c  ----  -----   ----------------------   ----   ----------  ---- ----  --
c   1.     1.    100.00000  1000.00000  f_excit               t     1
c   2.     2.     10.00000   100.00000  f_excit               t     1

c
c  code     : adas810
c  producer : h.p.summers
```

```
c  date     : 05/03/2002
c
c-------------------------------------------------------------------------------
```

A.13 *adf42*: driver data sets for ADAS810 emissivity calculations

A.14 *adf46*: driver data sets for BBGP for dielectronic recombination

The data sets contain atomic structure, threshold partial wave collisional strengths, transition probabilities, static dipole polarisabilities, quantum defect expansions, representative level and plasma specifications. These are sufficient to enable calculation of state selective dielectronic coefficients to bundle and bundle-nl shells of a complex ion in the *bbgp* approximation including the effects of doubly excited state re-distribution.

Data mnemonic:
Data root: **/home/adas/adas/adf46/**
Last update: Feb 13, 2009
Utilising subroutines: **run_adas708.for**

Formatted files to adf46 specification:

Iso-elec. Library	Members	Resol.	Comments	Quality
bbgp09#he	ls#li1 - ls#w71	ls	adas	Medium
bbgp09#li	ls#be1 - ls#w70	ls	adas	Medium
. . .				
bbgp09#ta	ls#w1	ls	adas	Medium

Notes:

Data lines:

Data lines	Format
seq ='SEQ' , nucchg =IZ0 , type ='CTYPE' , ADF46	1a80: keywords - a2,i,a2,a5
<blank line>	1a80
c10 , bwni = BWNO_I npri = NLVL_I	1a80: a10,keywords -f12,i3
[c10 has form 'aa+nn aaaa' where aa = elem. symb., nn= initial ion charge and aaaa = conf/term/levl if CTYPE=ca/ls/ic] [characters are left justified and numbers right justified]	
<underlines>	1a80
<descriptive headers>	1a80
<underlines>	1a80
endfor	
for i=1,NLVL_F	
IA_I(i) , CODE_I(i) , CSTRGA_I(i) , cnn , WA_I(i)	i5,a1,a,a,f13
[cnn has the form '(ISA_I(i))ILA_I(i)(XJA_I(i))' if CTYPE=ls/ic] and cnn has the form '(XJA_I(i))' if CTYPE=ca]	
endfor	
<blank line>	1a80

c10 , bwni = BWNO_F c4 = NLVL_F	1a80: a10,keywords -f12,i3
[c10 has form 'aa+nn aaaa' where aa = elem. symb., nn= initial ion charge and aaaa = conf/term/levl if CTYPE=ca/ls/ic]	
[c4 has form 'aaaa' where aaaa = ncfg/ntrm/nlvl if CTYPE=ca/ls/ic]	
<underlines>	1a80
<descriptive headers>	1a80
<underlines>	1a80
for i=1,NLVL_I	
IA_F(i) , CODE_F(i) , CSTRGA_F(i) , cnn , WA_F(i)	i5,a1,a,a,f13
[cnn has the form '(ISA_F(i))ILA_F(i)(XJA_F(i))' if CTYPE=ls/ic and cnn has the form '(XJA_F(i))' if CTYPE=ca]	
<blank line>	1a80
<block header>	1a80: keyword
[In normal order = 'transition parameters' (keyword: 'transition')	
<underlines>	1a80
c35,c221	a35,a221
[c35 is blank filled. c223 has the form 'l=0 l=1 .....l=NOMGL-1' where NOMGL is the number of partial waves present in the following table]	
<underlines>	1a256
for i=1,NTRN	
IDXL(i) , IDXU(i) , ITYP(i), AUL(i) , (OMGL(j,i)j=1,NOMGL)	3i5,5x,d10.2,22(f10.5)
endfor	
<blank line>	1a80
<block header>	1a80: keyword
[In normal order = 'polarisability ...' (keyword: 'polarisability')	
<underlines>	1a80
c30,c226	a30,a226
[c35 is descriptive . c223 has the form 'l=0 l=1 .....l=LMAX_QD' where LMAX_QD is the number of quantum defect expansions in the following table]	
<descriptive headers>	1a80
<underlines>	1a80
for i=1,NPOL	
IDXP(i) , DPOL(i) , (QD1(j,i),QD2(j,i),j=1,l=LMAX_QD+1)	i5,5x,23(f10.5)
endfor	
<blank line>	1a80
<block header>	1a80: keyword
[In normal order = 'representative ...' (keyword: 'representative')	
<underlines>	1a80
c11, NL1	*
[c11 has the form ' nl1 =']	
c11, NL2	*
[c11 has the form ' nl2 =']	
c11, NL3	*
[c11 has the form ' nl3 =']	
c11, c245	a11,a245
[c11 has the form ' nrep =. c245 has the form	

```
        (NREP(i),i=1,INREP) as blank separated integers]
c11, c245                                                    a11,a245
    [c11 has the form ' lrep =’. c245 has the form
        (LREP(i),i=1,ILREP) as blank separated integers]
<blank line>                                                 1a80
<block header>                                               1a80: keyword
    [In normal order = 'plasma ...' (keyword: 'plasma')
<underlines>                                                 1a80
c11, c245                                                    a11,a245
    [c11 has the form ' te =’. c245 has the form
        (TE(i),i=1,NTE) as blank separated real numbers ]
c11, c245                                                    a11,a245
    [c11 has the form ' dens =’. c245 has the form
        (DENS(i),i=1,NDENS) as blank separated real numbers ]
c11, c245                                                    a11,a245
    [c11 has the form ' tp =’. c245 has the form
        (TP(i),i=1,NTP) as blank separated real numbers ]
c11, c245                                                    a11,a245
    [c11 has the form ' densp =’. c245 has the form
        (DENSP(i),i=1,NDENSP) as blank separated real numbers ]
c11, ZP                                                      *
    [c11 has the form ' zp =' ]
c11, AMSP                                                    *
    [c11 has the form ' amsp =' ]
c11, c246                                                    a11,a245
    [c11 has the form 'c———-’. c245 is conventionally
        69 dashes '-' ]
```

Variable identification:

Name	Meaning	Comment
SEQ	iso-electronic sequence	two characters, blank fill to right
IZ0	nuclear charge	
CTYPE	ca, ls or ic denoting resolution	two characters
BWNO_I	ionisation potential of initial state ion (cm-1)	
NLVL_I	no. of initial ion configs, terms or levels	depending on resolution
IA_I()	index of initial states	
CODE_I()	code for metastables of initial state ion	* indicates a metastable
CSTRGA_I()	configuration string of initial states	standard notation
ISA_I()	multiplicity	only for ls and ic resolution
ILA_I()	total orbital quantum number	only for ls and ic resolution
XJA_I()	(stat. wt.-1)/2 or J	J only for ic resolution
BWNO_F	ionisation potential of final state ion (cm-1)	
NLVL_F	no. of final ion configs, terms or levels	depending on resolution

IA_F()	index of final states	
CODE_F()	code for metastables of final state ion	* indicates a metastable
CSTRGA_F()	configuration string of final states	standard notation
ISA_F()	multiplicity	only for ls and ic resolution
ILA_F()	total orbital quantum number	only for ls and ic resolution
XJA_F()	(stat. wt.-1)/2 or J	J only for ic resolution
NTRN	number of transitions	
IDXL()	index of lower parent ion level of trans	
IDXU()	index of upper parent ion level of trans	
ITYP()	type of parent transition (1=dipol, 2=non-dipol, 3=spin change)	
AUL()	transition probabilities (s-1)	
NOMGL()	number threshold partial wave collision strengths for parent transitions	
OMGL(,)	threshold partial wave collision strengths for parent transitions	
NPOL	number of polarisabilies	
IDXP()	index of parent ion level	
DPOL()	dipole polarisability of parent level	
LMAX_QD	largest quantum defect l-series expansion	
QDA1(,)	quantum defect expansion coefficient a0	
QDA2(,)	quantun defect expansion coefficient a1	
NL1	lowest n-shell of recombined ion for DR	
NL2	lowest l-resolved shell of recombined ion for DR	
NL3	highest bundle-n n-shell for recombined ion for dr	
INREP	number of representative n-shell	
NREP()	representative n-shells	
ILREP	number of representative l-shells	
LREP()	representative l-shells	
NTE	number of electron temperatures	
TE()	electron temperatures (K)	
NDENS	number of electron densities	
DENS()	electron densities (cm-3)	
NTP	number of positive ion temperatures	
TP()	positive ion temperatures (K)	
NDENSP	number of positive ion densities	
DENSP()	positive ion densities (cm-3)	
ZP	projectile ion effective charge	
AMSP	projectile ion effective mass number	

Sample 1: **/home/adas/adas/adf46/bbgp09#b/ls#o4.dat**

```
seq ='b'     nucchg = 8    type ='ls'                                        adf46

  o + 4 term indexing        bwnf =   6865884.2     nprf =  4
  -------------------
  indi    code                    S L   WJ       wno
  ----    ----                    - -   --  ------------
    1*  1s2 2s2                 (1)0( 1.0)          0.0
    2#  1s2 2s1 2p1             (1)1( 3.0)     166950.0
    3#  1s2 2s1 3s1             (1)0( 1.0)     561697.0
    4#  1s2 2s1 3p1             (1)1( 3.0)     580766.0

  o + 3 term indexing        bwni =   6092695.1     ntrm =  1
  -------------------
  indf    code                    S L   WJ       wno
  ----    ----                    - -   --  ------------
    1*  1s2 2s2 2p1             (2)1( 2.5)        257.3

transition parameters
---------------------
  idxl idxu typ_trn     aul      omgl(E=0)
  ---- ---- -------     ---      ---------
                                   l=0       l=1       l=2       l=3       l=4       l=5
                                   ---       ---       ---       ---       ---       ---
    1    4    1       2.31D+10   0.00599   0.00079   0.01329   0.21682   0.57813   0.00000
    3    4    1       1.67D+07   0.93985   1.02455   0.53798   0.46717   0.85458   0.00000
    2    4    2       1.00D+00   0.13598   0.03529   0.07155   0.01967   0.00039   0.00000

polarisability and quantum defect parameters
--------------------------------------------
  idxp       dpol             l=0                 l=1                 l=2
  ----       ----             ---                 ---                 ---
                           qd1       qd2       qd1       qd2       qd1       qd2
                           ---       ---       ---       ---       ---       ---
    1        0.99590   0.33845   0.06300   0.17670   0.07300   0.03874  -0.08300
    2        2.48760   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
    3       31.53780   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
    4       20.28190   0.00000   0.20000   0.00000   0.00000   0.00000   0.00000

representative n-shells
-----------------------
 nl1      =   2
 nl2      =  10
 nl3      = 900
 nrep     = 1 2 3 4 5 6 7 8 9 10 11 12 15 18 20 25 30 35 40 50 60 70 80 100 150 200 300 500 700 900
 lrep     = 0 1 2 3 4 5 6 7 8 9

plasma parameters
-----------------
 te       =  1.60E+04  2.00D+04  3.20D+04  5.00D+04  8.00D+04  1.00D+05  1.60D+05  3.20D+05
 ne       =  1.00E-04  2.00D-04  3.20D-04  5.00D-04  8.00D-04  1.00D-03  1.60D-03  3.20D-03
 tp       =  1.60E+04  2.00D+04  3.20D+04  5.00D+04  8.00D+04  1.00D+05  1.60D+05  3.20D+05
 np       =  1.00E-04  2.00D-04  3.20D-04  5.00D-04  8.00D-04  1.00D-03  1.60D-03  3.20D-03
 zp       = 1.0
 amsp     = 1836.0
```

```
c-------------------------------------------------------------------------------
c
c  Notes:   (1) te, tp in Kelvin; ne, np in cm^-3.
c           (2) tp and np relate to redistributive collisions with plasma ions
c               of charge zefp and atomic mass number amp.
c
c  Code     : adas701
c  Author   : Hugh Summers
c  Source   : /home/summers/adas_dev/adas/adf55/.dat
c  Date     : 05-02-2009
c
c-------------------------------------------------------------------------------
```

A.15 *adf48*: state selective radiative recombination coefficients

A.16 *adf54*: general promotional rule sets

Each data set of promotional rules provides a rule specification for all possible ground states of ions of elements.The central ADAS data base has generic samples for small, medium and large computer systems. In practice the rules would be customised for a particular computer system (see section 2.4. Customising is done for one element at a time, and it may be convenient for the user to archive the customised data set by element. The central ADAS data base has some such customised promotion rules data sets for EFDA-JET. There is potential redundancy in such archiving, since the rules for one element may well be apposite for all elements lighter than it.

Data mnemonic:
Data root: **/home/adas/adas/adf54/**
Last update: Aug 12, 2008
Utilising subroutines: **read_adf54.pro**

Formatted files to adf54 specification:

Element	Dataset	Sizing	Comments
	promotional_rules_small.dat	300 levels	
	promotional_rules_medium.dat	1000 levels	
	promotional_rules_large.dat	10000 levels	
	promotional_rules_custom.dat	*	
<elem. symbol>	promotional_rules_<elem. Symbol>.dat	1000 levels	

Data lines:

Data lines	Format
nmax, nel	31x,1i3,5x,1i2
do i=0 to n_index-1	
do j=0 to nel-1	
index(i*nel+j), config(i*nel+j)	1i5,1a*
enddo	
index[nel*i:(nel*(i+1))-1]	16x,10i5
n_el[nel*i:(nel*(i+1))-1]	16x,10i5
no_v_shl[nel*i:(nel*(i+1))-1]	16x,10i5
max_dn_v1[nel*i:(nel*(i+1))-1]	16x,10i5
min_dn_v1[nel*i:(nel*(i+1))-1]	16x,10i5
max_dl_v1[nel*i:(nel*(i+1))-1]	16x,10i5
min_dl_v1[nel*i:(nel*(i+1))-1]	16x,10i5
max_dn_v2[nel*i:(nel*(i+1))-1]	16x,10i5
min_dn_v2[nel*i:(nel*(i+1))-1]	16x,10i5
max_dl_v2[nel*i:(nel*(i+1))-1]	16x,10i5
min_dl_v2[nel*i:(nel*(i+1))-1]	16x,10i5
prom_cl[nel*i:(nel*(i+1))-1]	16x,10i5
max_n_cl[nel*i:(nel*(i+1))-1]	16x,10i5

```
      min_n_cl[nel*i:(nel*(i+1))-1]      16x,10i5
      max_l_cl[nel*i:(nel*(i+1))-1]      16x,10i5
      min_l_cl[nel*i:(nel*(i+1))-1]      16x,10i5
      max_dn_cl[nel*i:(nel*(i+1))-1]     16x,10i5
      min_dn_cl[nel*i:(nel*(i+1))-1]     16x,10i5
      max_dl_cl[nel*i:(nel*(i+1))-1]     16x,10i5
      min_dl_cl[nel*i:(nel*(i+1))-1]     16x,10i5
      fill_n_v1[nel*i:(nel*(i+1))-1]     16x,10i5
      fill_par[nel*i:(nel*(i+1))-1]      16x,10i5
      for_tr_sel[nel*i:(nel*(i+1))-1]    16x,10i5
      last_4f[nel*i:(nel*(i+1))-1]       16x,10i5
      grd_cmplx[nel*i:(nel*(i+1))-1]     16x,10i5
repeat
```

Variable identification:

Name	Meaning	Comment
nmax	number of ground configs.	
nel	number of ground configs per block	
n_index	number of blocks (= nmax/nel)	
index[]	ground config. index	
config[]	ground config. strinds	Cowan form
index[] n_el[]	number of electron	
no_v_shl[]	number of open shell	
max_dn_v1[]	maximum delta n promotion for first	
min_dn_v1[]	minimum delta n promotion for first	
max_dl_v1[]	maximum delta l promotion for first	
min_dl_v1[]	minimum delta l promotion for first	
max_dn_v2[]	maximum delta n promotion for second	
min_dn_v2[]	minimum delta n promotion for second	
max_dl_v2[]	maximum delta l promotion for second	
min_dl_v2[]	minimum delta l promotion for second	
prom_cl[]	promote from inner shell closed shells	
max_n_cl[]	maximum inner shell n for promotion from	
min_n_cl[]	minimum inner shell n for promotion from	
max_l_cl[]	maximum inner shell l for promotion from	
min_l_cl[]	minimum inner shell l for promotion from	
max_dn_cl[]	maximum delta n promotion for inner shell to	
min_dn_cl[]	minimum delta n promotion for inner shell to	
max_dl_cl[]	maximum delta l promotion for inner shell to	
min_dl_cl[]	minimum delta l promotion for inner shell to	
fill_n_v1[]	add all nl configurations	
fill_par[]	add parity	
for_tr_sel[]	Cowan option for radiative transitions	
last_4f[]	shift an electron valence shell to 4f	
grd_cmplx[]	include configurations of same complex	

Sample: **/home/adas/adas/adf54/promotional_rules_large.dat**

```
/ADF54    /PROMOTION RULES    /180    /10
index    config
    0    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p6
    1    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p5
    2    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p4
    3    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p3
    4    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p2
    5    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p1
    6    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2
    7    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s1
    8    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10
    9    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d9  5f0  5g0  6s1

index             0     1     2     3     4     5     6     7     8     9
n_el             86    85    84    83    82    81    80    79    78    78
no_v_shl          2     2     2     2     2     2     1     1     1     2
max_dn_v1         0     0     0     0     0     0     0     0     0     0
min_dn_v1        -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
max_dl_v1         4     4     4     4     4     4     4     4     4     4
min_dl_v1         0     0     0     0     0     0     0     0     0     0
max_dn_v2         0     0     0     0     0     0     0     0     0     1
min_dn_v2        -1    -1    -1    -1    -1    -1     0     0     0     0
max_dl_v2         3     3     3     3     3     3     0     0     0     2
min_dl_v2        -1    -1    -1    -1    -1    -1     0     0     0    -2
prom_cl           1     1     1     1     1     1     1     1     1     1
max_n_cl          5     5     5     5     5     5     5     5     5     5
min_n_cl          5     5     5     5     5     5     5     5     4     5
max_l_cl          3     3     3     3     3     3     3     3     3     3
min_l_cl          0     0     0     0     0     0     0     0     0     0
max_dn_cl         1     1     1     1     1     1     1     1     1     1
min_dn_cl         0     0     0     0     0     0     0     0     0     0
max_dl_cl         4     4     4     4     4     4     4     4     4     4
min_dl_cl        -3    -3    -3    -3    -3    -3    -3    -3    -3    -3
fill_n_v1         0     0     0     0     0     0     0     0     0     0
fill_par          1     1     1     1     1     1     1     1     0     0
for_tr_sel        3     3     3     3     3     3     3     3     3     3
last_4f           0     0     0     0     0     0     0     0     0     0
grd_cmplx         0     0     0     0     0     0     0     0     0     0

         .     .     .

index    config
  170    1s2  2s2  2p6
  171    1s2  2s2  2p5
  172    1s2  2s2  2p4
  173    1s2  2s2  2p3
  174    1s2  2s2  2p2
  175    1s2  2s2  2p1
  176    1s2  2s2
  177    1s2  2s1
  178    1s2
  179    1s1
```

index	170	171	172	173	174	175	176	177	178	179
n_el	10	9	8	7	6	5	4	3	2	1
no_v_shl	1	1	1	1	1	1	1	1	1	1
max_dn_v1	2	2	2	2	2	2	2	2	3	3
min_dn_v1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
max_dl_v1	3	3	3	3	3	3	3	3	3	3
min_dl_v1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
max_dn_v2	0	0	0	0	0	0	0	0	0	0
min_dn_v2	0	0	0	0	0	0	0	0	0	0
max_dl_v2	0	0	0	0	0	0	0	0	0	0
min_dl_v2	0	0	0	0	0	0	0	0	0	0
prom_cl	1	1	1	1	1	1	1	1	0	0
max_n_cl	2	2	2	2	2	2	1	1	1	1
min_n_cl	2	2	2	2	2	2	1	1	5	5
max_l_cl	1	1	1	1	1	1	1	1	1	1
min_l_cl	0	0	0	0	0	0	0	0	1	1
max_dn_cl	1	1	1	1	1	1	1	1	0	0
min_dn_cl	0	0	0	0	0	0	0	0	0	0
max_dl_cl	2	1	1	1	1	1	1	1	0	0
min_dl_cl	0	0	0	0	0	0	0	0	0	0
fill_n_v1	1	1	1	1	1	1	1	1	1	1
fill_par	0	0	0	0	0	0	0	0	0	0
for_tr_sel	3	3	3	3	3	3	3	3	3	3
last_4f	0	0	0	0	0	0	0	0	0	0
grd_cmplx	0	0	1	1	1	1	1	0	0	0

```
C-------------------------------------------------------------------------
C Generated from old adas8xx_promotion_rules_large.pro. This is a hand
C selected set of rules chosen by H. Summers.
C
C User:  Adam Foster
C Date:  15/02/08
C-------------------------------------------------------------------------
```

A.17 *adf55*: general dielectronic recombination promotional rules

Each data set of promotional rules provides a rule specification for all possible ground states of recombining ions of elements. The central ADAS data base has generic samples for small, medium and large computer systems. In practice the rules would be customised for a particular computer system (see section 2.4.

Data mnemonic:
Data root: **/home/adas/adas/adf55/**
Last update: Jan 29, 2009
Utilising subroutines: **read_adf55.pro**

Formatted files to adf55 specification:

Element	Dataset	Sizing	Comments
	promotional_rules_recom_small.dat	300 levels	
	promotional_rules_recom_medium.dat	1000 levels	
	promotional_rules_recom_large.dat	10000 levels	
	promotional_rules_recom_custom.dat	*	
<elem. symbol>	promotional_rules_<elem. Symbol>.dat	1000 levels	

Data lines:

Data lines	Format
nmax, nel	31x,1i3,5x,1i2
do i=0 to n_index-1	
do j=0 to nel-1	
index(i*nel+j), config(i*nel+j)	1i5,1a*
enddo	
index[nel*i:(nel*(i+1))-1]	16x,10i5
n_el[nel*i:(nel*(i+1))-1]	16x,10i5
no_v_shl[nel*i:(nel*(i+1))-1]	16x,10i5
max_dn_v1[nel*i:(nel*(i+1))-1]	16x,10i5
min_dn_v1[nel*i:(nel*(i+1))-1]	16x,10i5
max_dl_v1[nel*i:(nel*(i+1))-1]	16x,10i5
min_dl_v1[nel*i:(nel*(i+1))-1]	16x,10i5
max_dn_v2[nel*i:(nel*(i+1))-1]	16x,10i5
min_dn_v2[nel*i:(nel*(i+1))-1]	16x,10i5
max_dl_v2[nel*i:(nel*(i+1))-1]	16x,10i5
min_dl_v2[nel*i:(nel*(i+1))-1]	16x,10i5
prom_cl[nel*i:(nel*(i+1))-1]	16x,10i5
max_n_cl[nel*i:(nel*(i+1))-1]	16x,10i5

```
    min_n_cl[nel*i:(nel*(i+1))-1]      16x,10i5
    max_l_cl[nel*i:(nel*(i+1))-1]      16x,10i5
    min_l_cl[nel*i:(nel*(i+1))-1]      16x,10i5
    max_dn_cl[nel*i:(nel*(i+1))-1]     16x,10i5
    min_dn_cl[nel*i:(nel*(i+1))-1]     16x,10i5
    max_dl_cl[nel*i:(nel*(i+1))-1]     16x,10i5
    min_dl_cl[nel*i:(nel*(i+1))-1]     16x,10i5
    fill_n_v1[nel*i:(nel*(i+1))-1]     16x,10i5
    fill_par[nel*i:(nel*(i+1))-1]      16x,10i5
    for_tr_sel[nel*i:(nel*(i+1))-1]    16x,10i5
    last_4f[nel*i:(nel*(i+1))-1]       16x,10i5
    grd_cmplx[nel*i:(nel*(i+1))-1]     16x,10i5
    n_target[nel*i:(nel*(i+1))-1]      16x,10i5
    lmax_target[nel*i:(nel*(i+1))-1]   16x,10i5
repeat
```

Variable identification:

Name	Meaning	Comment
nmax	number of ground configs [refers to parent, that is recombining ion above and all following]	
nel	number of ground configs per block	
n_index	number of blocks (= nmax/nel)	
index[]	ground config. index	
config[]	ground config. strinds	Cowan form
index[] n_el[]	number of electron	
no_v_shl[]	number of open shell	
max_dn_v1[]	maximum delta n promotion for first	
min_dn_v1[]	minimum delta n promotion for first	
max_dl_v1[]	maximum delta l promotion for first	
min_dl_v1[]	minimum delta l promotion for first	
max_dn_v2[]	maximum delta n promotion for second	
min_dn_v2[]	minimum delta n promotion for second	
max_dl_v2[]	maximum delta l promotion for second	
min_dl_v2[]	minimum delta l promotion for second	
prom_cl[]	promote from inner shell closed shells	
max_n_cl[]	maximum inner shell n for promotion from	
min_n_cl[]	minimum inner shell n for promotion from	
max_l_cl[]	maximum inner shell l for promotion from	
min_l_cl[]	minimum inner shell l for promotion from	
max_dn_cl[]	maximum delta n promotion for inner shell to	
min_dn_cl[]	minimum delta n promotion for inner shell to	
max_dl_cl[]	maximum delta l promotion for inner shell to	
min_dl_cl[]	minimum delta l promotion for inner shell to	
fill_n_v1[]	add all nl configurations	
fill_par[]	add parity	
for_tr_sel[]	Cowan option for radiative transitions	
last_4f[]	shift an electron valence shell to 4f	
grd_cmplx[]	include configurations of same complex	
n_target[]	high spectator n-shell for Auger calcs. [refers to recombined ion above and following]	
lmax_target[]	largest spectator l-shell for Auger calcs.	

Sample: **/home/adas/adas/adf55/promotional_rules_large.dat**

```
/ADF55    /PROMOTION RULES    /180    /10
index    config
    0    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p6
    1    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p5
    2    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p4
    3    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p3
    4    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p2
    5    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p1
    6    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2
    7    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s1
    8    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10
    9    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d9  5f0  5g0  6s1

index             0     1     2     3     4     5     6     7     8     9
n_el             86    85    84    83    82    81    80    79    78    78
no_v_shl          2     2     2     2     2     2     1     1     1     2
max_dn_v1         0     0     0     0     0     0     0     0     0     0
min_dn_v1        -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
max_dl_v1         4     4     4     4     4     4     4     4     4     4
min_dl_v1         0     0     0     0     0     0     0     0     0     0
max_dn_v2         0     0     0     0     0     0     0     0     0     1
min_dn_v2        -1    -1    -1    -1    -1    -1     0     0     0     0
max_dl_v2         3     3     3     3     3     3     0     0     0     2
min_dl_v2        -1    -1    -1    -1    -1    -1     0     0     0    -2
prom_cl           1     1     1     1     1     1     1     1     1     1
max_n_cl          5     5     5     5     5     5     5     5     5     5
min_n_cl          5     5     5     5     5     5     5     5     4     5
max_l_cl          3     3     3     3     3     3     3     3     3     3
min_l_cl          0     0     0     0     0     0     0     0     0     0
max_dn_cl         1     1     1     1     1     1     1     1     1     1
min_dn_cl         0     0     0     0     0     0     0     0     0     0
max_dl_cl         4     4     4     4     4     4     4     4     4     4
min_dl_cl        -3    -3    -3    -3    -3    -3    -3    -3    -3    -3
fill_n_v1         0     0     0     0     0     0     0     0     0     0
fill_par          1     1     1     1     1     1     1     1     0     0
for_tr_sel        3     3     3     3     3     3     3     3     3     3
last_4f           0     0     0     0     0     0     0     0     0     0
grd_cmplx         0     0     0     0     0     0     0     0     0     0
n_target        100   100   100   100   100   100   100   100   100   100
lmax_target      10    10    10    10    10    10    10    10    10    10

         .     .     .

index    config
  170    1s2  2s2  2p6
  171    1s2  2s2  2p5
  172    1s2  2s2  2p4
  173    1s2  2s2  2p3
  174    1s2  2s2  2p2
  175    1s2  2s2  2p1
  176    1s2  2s2
  177    1s2  2s1
  178    1s2
```

```
  179    1s1

index           170   171   172   173   174   175   176   177   178   179
n_el             10     9     8     7     6     5     4     3     2     1
no_v_shl          1     1     1     1     1     1     1     1     1     1
max_dn_v1         2     2     2     2     2     2     2     2     3     3
min_dn_v1        -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
max_dl_v1         3     3     3     3     3     3     3     3     3     3
min_dl_v1        -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
max_dn_v2         0     0     0     0     0     0     0     0     0     0
min_dn_v2         0     0     0     0     0     0     0     0     0     0
max_dl_v2         0     0     0     0     0     0     0     0     0     0
min_dl_v2         0     0     0     0     0     0     0     0     0     0
prom_cl           1     1     1     1     1     1     1     1     0     0
max_n_cl          2     2     2     2     2     2     1     1     1     1
min_n_cl          2     2     2     2     2     2     1     1     5     5
max_l_cl          1     1     1     1     1     1     1     1     1     1
min_l_cl          0     0     0     0     0     0     0     0     1     1
max_dn_cl         1     1     1     1     1     1     1     1     0     0
min_dn_cl         0     0     0     0     0     0     0     0     0     0
max_dl_cl         2     1     1     1     1     1     1     1     0     0
min_dl_cl         0     0     0     0     0     0     0     0     0     0
fill_n_v1         1     1     1     1     1     1     1     1     1     1
fill_par          0     0     0     0     0     0     0     0     0     0
for_tr_sel        3     3     3     3     3     3     3     3     3     3
last_4f           0     0     0     0     0     0     0     0     0     0
grd_cmplx         0     0     1     1     1     1     1     0     0     0
n_target        100   100   100   100   100   100   100   100   100   100
lmax_target      10    10    10    10    10    10    10    10    10    10

C--------------------------------------------------------------------------
C Generated from old adas8xx_promotion_rules_large.pro. This is a hand
C selected set of rules chosen by H. Summers.
C
C User:  Hugh Summers
C Date:  29/01/09
C--------------------------------------------------------------------------
```

A.18 *adf56*: general ionisation promotional rules

Each data set of promotional rules provides a rule specification for all possible ground states of ionising ions of elements and an associated set of singly excited states based on a parent. The central ADAS data base has generic samples for small, medium and large computer systems. In practice the rules might be customised for a particular computer system (see section 2.4.

Data mnemonic:
Data root: **/home/adas/adas/adf56/**
Last update: Jan 09, 2009
Utilising subroutines: **read_adf56.pro**

Formatted files to adf56 specification:

Element	Dataset	Sizing	Comments
	promotional_rules_ionis_small.dat	grd drct	
	promotional_rules_ionis_medium.dat	grd drct & exca	
	promotional_rules_ionis_large.dat	exst & grd drct & exca	
<elem. symbol>	promotional_rules_ionis_<elem. Symbol>.dat	grd drct & exca	

Data lines:

Data lines	Format
nmax, nel	31x,1i3,5x,1i2
do i=0 to n_index-1	
do j=0 to nel-1	
index(i*nel+j), config(i*nel+j)	1i5,1a*
enddo	
index[nel*i:(nel*(i+1))-1]	16x,10i5
n_el[nel*i:(nel*(i+1))-1]	16x,10i5
no_v_shl[nel*i:(nel*(i+1))-1]	16x,10i5
v1_shl[nel*i:(nel*(i+1))-1]	16x,10i5
v2_shl[nel*i:(nel*(i+1))-1]	16x,10i5
drct_eval_v[nel*i:(nel*(i+1))-1]	16x,10i5
drct_eval_cl[nel*i:(nel*(i+1))-1]	16x,10i5
min_shl_cl[nel*i:(nel*(i+1))-1]	16x,10i5
exca_eval_v2[nel*i:(nel*(i+1))-1]	16x,10i5
max_dn_v2[nel*i:(nel*(i+1))-1]	16x,10i5
min_dn_v2[nel*i:(nel*(i+1))-1]	16x,10i5
max_dl_v2[nel*i:(nel*(i+1))-1]	16x,10i5

```
      min_dl_v2[nel*i:(nel*(i+1))-1]          16x,10i5
      exca_eval_cl[nel*i:(nel*(i+1))-1]       16x,10i5
      max_dn_cl[nel*i:(nel*(i+1))-1]          16x,10i5
      min_dn_cl[nel*i:(nel*(i+1))-1]          16x,10i5
      max_dl_cl[nel*i:(nel*(i+1))-1]          16x,10i5
      min_dl_cl[nel*i:(nel*(i+1))-1]          16x,10i5
      exst_eval [nel*i:(nel*(i+1))-1]         16x,10i5
      exst_adf00_prt[nel*i:(nel*(i+1))-1]     16x,10i5
      exst_prt_hole_shl[nel*i:(nel*(i+1))-1]  16x,10i5
      max_n_exst[nel*i:(nel*(i+1))-1]         16x,10i5
      max_l_exst[nel*i:(nel*(i+1))-1]         16x,10i5
      drct_eval_exst_v[nel*i:(nel*(i+1))-1]   16x,10i5
      drct_eval_exst_cl[nel*i:(nel*(i+1))-1]  16x,10i5
      exca_eval_exst_v[nel*i:(nel*(i+1))-1]   16x,10i5
      exca_eval_exst_cl[nel*i:(nel*(i+1))-1]  16x,10i5
repeat
```

Variable identification:

Name	Meaning	Comment
nmax	number of ground configs.	
nel	number of ground configs per block	
n_index	number of blocks (= nmax/nel)	
index[]	ground config. index	
config[]	ground config. strings	Cowan form
n_el[]	number of electrons	
no_v_shl[]	number of shells to treat as valence shells. Max. 2 relevant to relating ion and parent	
v1_shl[]	first valence shell position in adf56 configuration specifications.	
v2_shl[]	second valence shell position in adf56 configuration specifications. zero if none defined.	
drct_eval_v[]	evaluate direct ionisation from the valence shell(s).	
drct_eval_cl[]	evaluate direct ionisation from other non-valence (closed) shells.	
min_shl_cl[]	lowest closed shell to include (position in adf56 configuration specifications).	
exca_eval_v2[]	evaluate excitation/autoionisation from second valence shell if identified.	
max_dn_v2[]	maximum change in v2 n-shell to be included.	
min_dn_v2[]	minimum change in v2 n-shell to be include.	
max_dl_v2[]	maximum change in v2 l-shell to be included.	
min_dl_v2[]	minimum change in v2 l-shell to be include.	
exca_eval_cl[]	evaluate excitation/autoionisation from other non-valence (closed) shells.	
max_dn_cl[]	maximum change in closed n-shell to be included.	
min_dn_cl[]	minimum change in closed n-shell to be included.	
max_dl_cl[]	maximum change in closed l-shell to be included.	
min_dl_cl[]	minimum change in closed l-shell to be included.	
exst_eval[]	evaluate ionisation from excited states.	
exst_adf00_prt[]	assume parent for building excited states is as present in the adf00 data set for the ion.	
exst_prt_hole_shl[]	specify position of shell in ground configuration to form parent if not from adf00 above.	
max_n_exst[]	maximum n-shell for excited states to be included.	
max_l_exst[]	maximum l-shell for excited states to be included.	
drct_eval_exst_v[]	evaluate direct ionisation from excited state valence shells.	
drct_eval_exst_cl[]	evaluate direct ionisation from excited state non-valence (closed) shells.	
exca_eval_exst_v[]	evaluate excitation/autoionisation for excited states from valence shells (v1 and v2 above).	
exca_eval_exst_cl[]	evaluate excitation/autoionisation for excited states from non-valence (closed) shells.	

Sample: **/home/adas/adas/adf56/promotional_rules_ionis_large.dat**

```
/ADF56    /PROMOTION RULES    /180    /10
index    config
    0    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p6
    1    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p5
    2    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p4
    3    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p3
    4    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p2
    5    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2  6p1
    6    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s2
    7    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10 5f0  5g0  6s1
    8    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d10
    9    1s2  2s2  2p6  3s2  3p6  3d10 4s2  4p6  4d10 4f14 5s2  5p6  5d9  5f0  5g0  6s1

index             0     1     2     3     4     5     6     7     8     9
n_el             86    85    84    83    82    81    80    79    78    78
no_v_shl          2     2     2     2     2     2     1     1     1     2
v1_shl           17    17    17    17    17    17    16    16    13    16
v2_shl           16    16    16    16    16    16     0     0     0    13
drct_eval_v       1     1     1     1     1     1     1     1     1     1
drct_eval_cl      1     1     1     1     1     1     1     1     1     1
min_shl_cl       11    11    11    11    11    11    11    11    11    11
exca_eval_v2      1     1     1     1     1     1     1     1     1     1
max_dn_v2         0     0     0     0     0     0     0     0     0     1
min_dn_v2        -1    -1    -1    -1    -1    -1     0     0     0     0
max_dl_v2         3     3     3     3     3     3     0     0     0     2
min_dl_v2        -1    -1    -1    -1    -1    -1     0     0     0    -2
exca_eval_cl      1     1     1     1     1     1     1     1     1     1
max_dn_cl         1     1     1     1     1     1     1     1     1     1
min_dn_cl         0     0     0     0     0     0     0     0     0     0
max_dl_cl         4     4     4     4     4     4     4     4     4     4
min_dl_cl        -3    -3    -3    -3    -3    -3    -3    -3    -3    -3
exst_eval         1     1     1     1     1     1     1     1     1     1
exst_adf00_prt    1     1     1     1     1     1     1     1     1     1
exst_prt_hole_shl 0     0     0     0     0     0     0     0     0     0
max_n_exst        0     0     0     0     0     0     0     0     0     0
max_l_exst        0     0     0     0     0     0     0     0     0     0
drct_eval_exst_v  1     1     1     1     1     1     1     1     1     1
drct_eval_exst_cl 1     1     1     1     1     1     1     1     1     1
exca_eval_exst_v  1     1     1     1     1     1     1     1     1     1
exca_eval_exst_cl 1     1     1     1     1     1     1     1     1     1

         .     .     .

index    config
  170    1s2  2s2  2p6
  171    1s2  2s2  2p5
  172    1s2  2s2  2p4
  173    1s2  2s2  2p3
  174    1s2  2s2  2p2
  175    1s2  2s2  2p1
  176    1s2  2s2
  177    1s2  2s1
  178    1s2
```

```
  179    1s1

index           170   171   172   173   174   175   176   177   178   179
n_el             10     9     8     7     6     5     4     3     2     1
no_v_shl          1     1     1     1     1     1     1     1     1     1
v1_shl           17    17    17    17    17    17    16    16    13    16
v2_shl           16    16    16    16    16    16     0     0     0    13
drct_eval_v       1     1     1     1     1     1     1     1     1     1
drct_eval_cl      1     1     1     1     1     1     1     1     1     1
min_shl_cl       11    11    11    11    11    11    11    11    11    11
exca_eval_v2      1     1     1     1     1     1     1     1     1     1
max_dn_v2         0     0     0     0     0     0     0     0     0     1
min_dn_v2        -1    -1    -1    -1    -1    -1     0     0     0     0
max_dl_v2         3     3     3     3     3     3     0     0     0     2
min_dl_v2        -1    -1    -1    -1    -1    -1     0     0     0    -2
exca_eval_cl      1     1     1     1     1     1     1     1     1     1
max_dn_cl         1     1     1     1     1     1     1     1     1     1
min_dn_cl         0     0     0     0     0     0     0     0     0     0
max_dl_cl         4     4     4     4     4     4     4     4     4     4
min_dl_cl        -3    -3    -3    -3    -3    -3    -3    -3    -3    -3
exst_eval         1     1     1     1     1     1     1     1     1     1
exst_adf00_prt    1     1     1     1     1     1     1     1     1     1
exst_prt_hole_shl 0     0     0     0     0     0     0     0     0     0
max_n_exst        0     0     0     0     0     0     0     0     0     0
max_l_exst        0     0     0     0     0     0     0     0     0     0
drct_eval_exst_v  1     1     1     1     1     1     1     1     1     1
drct_eval_exst_cl 1     1     1     1     1     1     1     1     1     1
exca_eval_exst_v  1     1     1     1     1     1     1     1     1     1
exca_eval_exst_cl 1     1     1     1     1     1     1     1     1     1

C--------------------------------------------------------------------------
C This is a hand  selected set of rules chosen by H. Summers.
C
C User:  Hugh Summers
C Date:  09/01/09
C--------------------------------------------------------------------------
```

Appendix B

IDL procedures

Procedure	Current location	Local checks			Central ADAS	
		Txt	Opr	Lnk	CVS	Rel
read_adf00.pro	/home/hps/adas_dev/idl/read_adf/	y	n	n	n	n
xxdtes.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
xxcftr.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
c5dplr.pro	/home/summers/adas_dev/idl/adas3xx/adas305/	y	n	n	n	n
read_adf15.pro	/home/adas/idl/adaslib/read_adf/	y	n	n	n	n
adas_vector.pro	/home/adas/idl/adaslib/util/	y	n	n	n	n
read_adf54.pro	/home/summers/adas_dev/idl/read_adf/	y	n	n	n	n
adas8xx_promotion_rules.pro	/home/hps/adas_dev/idl/adas8xx/adaslib/	y	n	n	n	n
adas8xx_promotions.pro	/home/hps/adas_dev/idl/adas8xx/adaslib/	y	n	n	n	n
cfg2occ.pro	/home/adas/idl/adaslib/atomic/	y	n	n	n	n
adas8xx_create_drivers.pro	/home/hps/adas_dev/idl/adas8xx/adaslib/	y	n	n	n	n
adas8xx_create_ca_adf04.pro	/home/hps/adas_dev/idl/adas8xx/adaslib/	y	n	n	n	n
adas8xx_create_ls_ic_adf04.pro	/home/hps/adas_dev/idl/adas8xx/adaslib/	y	n	n	n	n
run_adas808.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
adas8xx_opt_promotions_control.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
adas8xx_opt_expand_promotions.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
r8fbch.pro	/home/adas/idl/adaslib/atomic/	y	n	n	n	n
r8necip.pro	/home/adas/idl/adas2xx/adas208/	y	n	n	n	n
config_orbital_energies.pro	/home/adas/adaslib/atomic/	y	n	n	n	n
tev_alf_s.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
sbchid_cfg_tot.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
read_adf56.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
adas8xx_ionis_promotion_rules.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
adas8xx_ionis_promotions.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
adas8xx_ionis_create_drivers.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
adas8xx_ionis_create_ca_adf23.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
run_adas813.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
read_adf55.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n

Procedure	Current location	Local checks			Central ADAS	
		Txt	Opr	Lnk	CVS	Rel
alf_r_bdn.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
alf_r_bdnl.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
alf_r_tot.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
alf_d_bgf.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
alf_d_bgp.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
alf_d_bbgp.pro	/home/summers/adas_dev/idl/adaslib/atomic/	y	n	n	n	n
read_adf55.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
run_adas407.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
run_adas408.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
run_adas316.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
preview_natural_partition.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
run_adas416.pro	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n

read_adf00.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  read_adf00
;
; PURPOSE    :  Reads an adf00 element file from IDL
;               Called from IDL using the syntax
;                  read_adf00,file=..., z0=... etc
;
; ARGUMENTS  :  Arguments are non-positional named parameters.   All output
;               arguments will be defined appropriately.
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  file       I     str     full name of ADAS adf00 file
;               dir        I     str     adf00 directory
;               z0         I     int     atomic number
;               z_ion      I     int     Ion stage(s) of interest. (preferred)
;               z1         I     int     Ion stage(s) of interest. (deprecated)
;               [Specifying both z_ion and z1 is disallowed]
; OPTIONAL      ionpot     O     double  Ion potential(s).
;               config     O     double  Ground state configuration(s).
;
; KEYWORDS   :  /all  - return all ion stages. Note z1 must be an
;                         argument in this case.
;
; NOTES      : Either file or dir and z0 or z0 alone are required. If
;              only z0 is given then central ADAS adf00 is queried.
;
;
; AUTHOR     :  Martin O'Mullane
;
; DATE       :  07-02-2001
;
;
; MODIFIED:
;       1.2     Allan Whiteford
;                 - Added /help keyword
;       1.3     Hugh Summers
;                 - Added preferred keywords z_ion for z1. Deprecate z1
;                   and fault if both z1 and z_ion set.
;
; VERSION:
;       1.1    04-12-2002
;       1.2    07-04-2005
;       1.3    22-08-2008
;
;-
;-----------------------------------------------------------------------



PRO read_adf00, file    =  file,       $
                dir     =  dir,        $
```

```
                z0      =  z0,         $
                z1      =  z1,         $
                z_ion   =  z_ion,      $
                config  =  config,     $
                ionpot  =  ionpot,     $
                all     =  all,        $
                help    =  help

;-----------------------------------------------------------------------
```

Notes:

xxdtes.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  xxdtes
;
; PURPOSE    :  Detects if the configuration string from a specific ion
;               level list line is of Eissner or standard form.
;
; ARGUMENTS  :  All output arguments will be defined appropriately.
;               Inputs will be converted to correct type, if possible,
;               internally without changing calling type.
;
;                xxdtes, in_cfg = in_cfg, is_eissner=is_eissner
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  in_cfg     I     string  input configuration
;               is_eissner O     integer 1 = true if in_cfg is of Eissner form
;                                        0 = false
;
; KEYWORDS      None
;
;
; NOTES      :  Calls the fortran code. C returns weird stuff for true
;               and false when mapped to fortran logical variables
;               (-1 is true, 0 is false?)
;
;
;
; AUTHOR     :  Martin O'Mullane
;
; DATE       :  08-04-2002
;
;
; MODIFIED:
;       1.1     Martin O'Mullane
;               - First version.
;       1.2     Martin O'Mullane
;               - Add an interface fortran subroutine to turn logicals
;                 into integers. Should fix endian problems.
; 1.3 Allan Whiteford
```

```
; - Changed wrapper path to be just ADASFORT.
;
; VERSION:
;       1.1    08-04-2002
;       1.2    05-02-2003
;       1.3    10-08-2004
;-
;-----------------------------------------------------------------------

PRO xxdtes, in_cfg = in_cfg, is_eissner=is_eissner
```

Notes:

xxcftr.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  xxcftr
;
; PURPOSE    :  Converts a configuration character string, such as occurs
;               in a specific ion file level list, between Eissner and
;               standard forms.
;
; ARGUMENTS  :  All output arguments will be defined appropriately.
;               Inputs will be converted to correct type, if possible,
;               internally without changing calling type.
;
;                xxcftr, in_cfg = in_cfg, out_cfg=out_cfg, type=type
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  in_cfg     I     string  input configuration
;               out_cfg    I     string  output configuration
;               type       I     integer type of conversion
;                                   = 1 => standard form out, standard form in
;                                     2 => Eissner  form out, standard form in
;                                     3 => standard form out, Eissner  form in
;                                     4 => Eissner  form out, Eissner  form in
;
; KEYWORDS      None
;
;
; NOTES      :  Calls the fortran code.
;
;
; AUTHOR     :  Martin O'Mullane
;
; DATE       :  08-04-2002
;
;
; MODIFIED:
;       1.1     Martin O'Mullane
;               - First version.
```

```
; 1.3 Allan Whiteford
; - Changed wrapper path to be just ADASFORT.
;
; VERSION:
;       1.1    08-04-2002
;       1.1    10-08-2004
;-
;-----------------------------------------------------------------------------


PRO xxcftr, in_cfg = in_cfg, out_cfg=out_cfg, type=type
```

Notes:

c5dplr.pro

```
;-----------------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  c5dplr
;
; PURPOSE    :  Doppler broaden Stark components.
;
; ARGUMENTS  :  All output arguments will be defined appropriately.
;               Inputs will be converted to correct type, if possible,
;               internally without changing calling type.
;
;                c5dplr
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :
;               amss       I     double  Atomic mass of hydrogen in plasma
;
;               ndpix      I     long    Maximum number of pixels in range
;               npix       I     long    Actual number of pixels in range
;               wvmin      I     double  Minimum wavelength (A)
;               wvmax      I     double  Maximum wavelength (A)
;               ndcomp     I     long    Maximum number of components
;               ncomp      I     long    Actual number of components
;               wvcomp()   I     double  Wavelength of component
;               emcomp()   I     double  Emissivity of component
;               tev        I     double  Specific plasma electron temperature (eV)
;               amss       I     double  Atomic mass of hydrogen in plasma
;               doppler()  O     double  Doppler broadened feature
;
; NOTES      :  Calls fortran code.
;
;
; AUTHOR     :  Martin O'Mullane
;
; DATE       :  21/02/05
```

```
;
;
; MODIFIED:
;       1.1     Martin O'Mullane
;               - First version.
;
;
; VERSION:
;       1.1    21/02/05
;
;-
;-----------------------------------------------------------------------

PRO c5dplr, ndpix, npix, wvmin, wvmax, ndcomp,   $
            ncomp, wvcomp, emcomp, tev, amss,    $
            doppler

;-----------------------------------------------------------------------
```

Notes:

read_adf15.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  read_adf15
;
; PURPOSE    :  Reads adf15 (PEC) files from the IDL command line.
;               called from IDL using the syntax
;               read_adf15,file=...,block=...,te=... etc
;
; ARGUMENTS  :  All output arguments will be defined appropriately.
;
;               NAME      I/O    TYPE   DETAILS
; REQUIRED   :  file       I     str    full name of ADAS adf15 file
;               block      I     int    selected block
;               te         I     real() temperatures requested
;               dens       I     real() densities requested
; OPTIONAL      data       O      -     PEC data
;               wlngth     O      -     wavelength of transition
;               iz0        O     int    return guess of nuclear charge
;                                       (returns -1 if unable to guess)
;               izz        O     int    return guess of ion charge
;                                       (returns -1 if unable to guess)
;               iz1        O     int    return guess of ion charge+1
;                                       (returns -1 if unable to guess)
;
; KEYWORDS      all        I      -     if specified data is 2D of
;                                       temperature and density.
;               kelvin     I      -     requested temperature in K (default eV)
;
```

```
; NOTES      :  This is part of a chain of programs - read_adf15.c and
;               readadf15.for are required.
;
; AUTHOR     :  Martin O'Mullane
;
; DATE       :  20-07-99
;
; MODIFIED   :
;       1.2     Martin O'Mullane
;                 - No limit on number of Te/dens returned
;                 - Returns a 2D array if /all is specified.
;                 - Temperatures can be requested in K if /kelvin is specified.
;       1.3     Martin O'Mullane
;                 - Regularised comments
;       1.4     Allan Whiteford
;                 - Added in functionality to guess at nuclear and ion charge.
;       1.5     Allan Whiteford
;                 - Added /help keyword.
;       1.6     Martin O'Mullane
;                 - Add fulldata.
;       1.7     Martin O'Mullane
;                 - Increase nstore to 500.
;
; VERSION    :
;       1.1    20-07-1999
;       1.2    05-03-2001
;       1.3    15-03-2002
;       1.4    08-05-2003
;       1.5    07-04-2005
;       1.6    12-04-2005
;       1.7    22-04-2005
;-
;-----------------------------------------------------------------------

PRO read_adf15, file     = file,     $
                block    = block,    $
                te       = te,       $
                dens     = dens,     $
                data     = data,     $
                wlngth   = wlngth,   $
                all      = all,      $
                kelvin   = kelvin,   $
                iz0      = iz0,      $
                izz      = izz,      $
                iz1      = iz1,      $
                fulldata = fulldata, $
                help     = help
;-----------------------------------------------------------------------
```

Notes:

adas_vector.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT:
;       ADAS
;
; NAME:
;       ADAS_VECTOR
;
; PURPOSE:
;       Generate a vector of values given the minimum, maximum
;       and number of points.
;
; NOTES:
;       The default is to generate a log based distribution.
;
;
; INPUTS:
;       LOW    -  minimum value.
;       HIGH   -  maximum value.
;       NUM    -  number of points.
;
; OPTIONAL INPUTS:
;       None
;
; OUTPUTS:
;
; OPTIONAL OUTPUTS:
;       None.
;
; KEYWORD PARAMETERS:
;       LINEAR - if selected the distribution is linear.
;
; CALLS:
;       None
;
; SIDE EFFECTS:
;       None
;
; CATEGORY:
;       UNIX system IDL utility.
;
; WRITTEN:
;       Martin O'Mullane, 25-11-2004
;
;
; MODIFIED:
;       1.1     Martin O'Mullane
;               - First release.
;
; VERSION:
;       1.1     25-11-2004
;
;-
;-----------------------------------------------------------------------

FUNCTION adas_vector, low=low, high=high, num=num, linear=linear
```

```
;-----------------------------------------------------------------------
```

Notes:

read_adf54.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  read_adf54
;
; PURPOSE    :  Reads adf54 (promotion rules) files from the IDL command line.
;               called from IDL using the syntax
;               read_adf54,file=...,fulldata=fulldata
;
; ARGUMENTS  :  All output arguments will be defined appropriately.
;
;               NAME      I/O    TYPE   DETAILS
; REQUIRED   :  file       I     str    full name of ADAS adf15 file
;               fulldata   O     str    Structure containing the
;                                        adf54 details.
; NOTES      :
;
; The fulldata structure is defined:
;
;              config     : ground configuration
;              index      : reference index
;              n_el       : number of electron
;              no_v_shl   : number of open shell
;              max_dn_v1  : maximum delta n promotion for first
;              min_dn_v1  : minimum delta n promotion for first
;              max_dl_v1  : maximum delta l promotion for first
;              min_dl_v1  : minimum delta l promotion for first
;              max_dn_v2  : maximum delta n promotion for second
;              min_dn_v2  : minimum delta n promotion for second
;              max_dl_v2  : maximum delta l promotion for second
;              min_dl_v2  : minimum delta l promotion for second
;              prom_cl    : promote from inner shell closed shells
;              max_n_cl   : maximum inner shell n for promotion from
;              min_n_cl   : minimum inner shell n for promotion from
;              max_l_cl   : maximum inner shell l for promotion from
;              min_l_cl   : minimum inner shell l for promotion from
;              max_dn_cl  : maximum delta n promotion for inner shell to
;              min_dn_cl  : minimum delta n promotion for inner shell to
;              max_dl_cl  : maximum delta l promotion for inner shell to
;              min_dl_cl  : minimum delta l promotion for inner shell to
;              fill_n_v1  : add all nl configurations
;              fill_par   : add parity
;              for_tr_sel : Cowan option for radiative transitions
;              last_4f    : shift an electron valence shell to 4f
;              grd_cmplx  : include configurations of same complex
;
```

```
; AUTHOR     :  Alessandra Giunta
;
; DATE       :  02-10-2007
;
;
; MODIFIED:
;       1.1    Alessandra Giunta
;               - First release.
;
; VERSION:
;       1.1      02-10-2007
;-
;-----------------------------------------------------------------------

pro read_adf54, file = file, fulldata = fulldata

;-----------------------------------------------------------------------
```

Notes:

adas8xx_promotion_rules.pro

```
;-------------------------------------------------------------------------------
;+
;
; NAME     : adas8xx_promotion_rules
;
; PURPOSE  : To set the default promotional rules for configuration generation
;            of the complete set of ions of an element.
;
; CATEGORY : ADAS
;
; USE      : Usually called before the promotional routine 'default_promotions'
;
; INPUT    :
;
;   (I*4)   z0_nuc          = keyword = nuclear charge
;   (C* )   a54file         = keyword = adf54 promotion rules dataset
;
; OUTPUTS  :
;
;   (R*8)   ionpot[]        = keyword = ionis. potential (ev) for each ion of element
;                                       1st dim: ion charge 0 --> z0_nuc-1
;   (struc) rules           = keyword = structure of rules for ion charges 0 --> z0_nuc-1
;
; NOTES    : The rules structure is defined as (vectors span 0 --> z0_nuc-1)
;
;              index[]      : index of ground configuration of each ion of element in adf54 file
;              config[]     : ground configuration for each ion of element
;              n_el[]       : number of electrons for each ion of element
;              no_v_shl[]   : number of open (valence) shells. Include outer-most shell even if closed
;              max_dn_v1[]  : maximum delta n promotion for first (outer-most) valence shell
;              min_dn_v1[]  : minimum delta n promotion for first (outer-most) valence shell.
```

```
;                                       Negative value allows access to inner unoccupied or open shell
;              max_dl_v1[]  : maximum delta l promotion for first (outer-most) valence shell
;              min_dl_v1[]  : minimum delta l promotion for first (outer-most) valence shell.
;              max_dn_v2[]  : maximum delta n promotion for second (inner-most) valence shell
;              min_dn_v2[]  : maximum delta n promotion for second (inner-most) valence shell
;              max_dl_v2[]  : maximum delta l promotion for second (inner-most) valence shell
;              min_dl_v2[]  : minimum delta l promotion for second (inner-most) valence shell
;              prom_cl[]    : promote from inner shell closed shells (1=yes,0=no)
;              max_n_cl[]   : maximum inner shell n for promotion from
;              min_n_cl[]   : minimum inner shell n for promotion from
;              max_l_cl[]   : maximum inner shell l for promotion from
;              min_l_cl[]   : minimum inner shell l for promotion from
;              max_dn_cl[]  : maximum delta n promotion for inner shell to
;              min_dn_cl[]  : minimum delta n promotion for inner shell to
;              max_dl_cl[]  : maximum delta l promotion for inner shell to
;              min_dl_cl[]  : minimum delta n promotion for inner shell to
;                                       Negative value allows access to inner unoccupied or open shell
;              fill_n_v1[]  : add all nl configurations of outer valence shell n (1=yes,0=no)
;              fill_par[]   : if n_fill only add opposite parity to valence shell else add both pariti
;              for_tr_sel[] : Cowan option for radiative transitions 1 - first parity, 2 or 3(default)
;              last_4f[]    : shift an electron valence shell to unfilled 4f as extra ground
;              grd_cmplx[]  : include configurations of same complex as ground configuation for valenc
;
;
; ROUTINES:
;            NAME            TYPE      COMMENT
;            read_adf00      ADAS      reads an adf00 file
;            read_adf54      ADAS      reads an adf54 promotion rules dataset
;
; CATEGORY:
;       Adas system.
;
; WRITTEN:
;       H. P. Summers, University of Strathclyde,29-06-06.
;
; MODIFIED:
;       1.1     H. P. Summers
;                 - First version put into CVS.
;       1.2     A. Foster & H. P. Summers
;                 - Changed to use an adf54 data file with customised promotion rules
;       1.3     H. P. Summers
;                 - Changed order of parameters to put 'a54file' near the beginning,
;                   added index and introduced ref_rules and rules structures
;       1.4     H. P. Summers
;                 - Replaced variable name z0 by z0_nuc to avoid confusion
;;
; VERSION:
;       1.1   25-08-06
;       1.2   23-07-08
;       1.3   19-08-08
;       1.4   22-08-08
;
;-
;-----------------------------------------------------------------------------

PRO adas8xx_promotion_rules, z0_nuc     = z0_nuc,                           $
                             a54file    = a54file,                          $
```

```
                              ionpot     = ionpot,                              $
     rules      = rules

;-----------------------------------------------------------------------------
```

Notes:

adas8xx_promotions.pro

```
;-----------------------------------------------------------------------------
;+
;
; NAME     : adas8xx_promotions
;
; PURPOSE  : generate a set of excited configurations for an ion of an
;            element from an initial ground configuration and a set of rules.
;
; CATEGORY : ADAS
;
; USE      : Usually called after 'adas8xx_promotion_rules'
;
; INPUT    :
;
;   (I*4)   z0_nuc          = keyword = nuclear charge
;   (I*4)   z_ion           = keyword = selected ion charge
;   (R*8)   ionpot          = keyword = ionis. potential (ev) for an ion
;   (struc) rules           = keyword = structure of rules for ion charges 0 --> z0_nuc-1
;
;
; OUTPUTS   :
;
;   (C* )   grd_cfg         = keyword = ground configuration
;   (I*4)   grd_occ[]       = keyword = ground occupation nos.
;   (C* )   ex_cfg[]        = keyword = excited configurations
;   (I*4)   grd_par         = keyword = ground parity
;   (I*4)   ex_par[]        = keyword = excited parities
;   (I*4)   grd_zc_cow[]        = keyword = grd. eff. charge for Cowan adf34 driver
;                                       n.b. prescription ok for z0_nuc>19.  For
;                                            z0_nuc<19 revert to zc_cow=z_ion=ion charge+1
;   (I*4)   ex_zc_cow[]         = keyword = exc. eff. charge for Cowan adf34 driver
;                                       n.b. prescription ok for z0_nuc>19.  For
;                                            z0_nuc<19 revert to zc_cow=z_ion+1=ion charge+1
;   (I*4)   oc_store[]      = keyword = configuration occupancies
;   (I*4)   no_configs[]    = keyword = no. of configs in 7 categories (0: grd, 1: 1st val., 2: 2nd va
;                                       3: inner shell, 4: all val. n, 5: alter. grd., 6: grd complex.
;   (I*4)   no_terms []     = keyword = no. of levels in 7 categories (as above)
;   (I*4)   no_levels[]     = keyword = no. of levels in 7 categories (as above)
;
; NOTES    : The rules structure is defined as (vectors span 0 --> z0_nuc-1)
;
;              index[]      : index of ground configuration of each ion of element in adf54 file
;              config[]     : ground configuration for each ion of element
;              n_el[]       : number of electrons for each ion of element
```

```
;              no_v_shl[]   : number of open (valence) shells. Include outer-most shell even if closed
;              max_dn_v1[]  : maximum delta n promotion for first (outer-most) valence shell
;              min_dn_v1[]  : minimum delta n promotion for first (outer-most) valence shell.
;                                 Negative value allows access to inner unoccupied or open shell
;              max_dl_v1[]  : maximum delta l promotion for first (outer-most) valence shell
;              min_dl_v1[]  : minimum delta l promotion for first (outer-most) valence shell.
;              max_dn_v2[]  : maximum delta n promotion for second (inner-most) valence shell
;              min_dn_v2[]  : maximum delta n promotion for second (inner-most) valence shell
;              max_dl_v2[]  : maximum delta l promotion for second (inner-most) valence shell
;              min_dl_v2[]  : minimum delta l promotion for second (inner-most) valence shell
;              prom_cl[]    : promote from inner shell closed shells (1=yes,0=no)
;              max_n_cl[]   : maximum inner shell n for promotion from
;              min_n_cl[]   : minimum inner shell n for promotion from
;              max_l_cl[]   : maximum inner shell l for promotion from
;              min_l_cl[]   : minimum inner shell l for promotion from
;              max_dn_cl[]  : maximum delta n promotion for inner shell to
;              min_dn_cl[]  : minimum delta n promotion for inner shell to
;              max_dl_cl[]  : maximum delta l promotion for inner shell to
;              min_dl_cl[]  : minimum delta n promotion for inner shell to
;                                 Negative value allows access to inner unoccupied or open shell
;              fill_n_v1[]  : add all nl configurations of outer valence shell n (1=yes,0=no)
;              fill_par[]   : if n_fill only add opposite parity to valence shell else add both pariti
;              for_tr_sel[] : Cowan option for radiative transitions 1 - first parity, 2 or 3(default)
;              last_4f[]    : shift an electron valence shell to unfilled 4f as extra ground
;              grd_cmplx[]  : include configurations of same complex as ground configuation for valenc
;
; ROUTINES:
;            NAME              TYPE    COMMENT
;            get_parity        local   returns the parity given occupation numbers
;            new_occup         local   determine whether the occupation vector
;                                      matchs a group.
;            write_config      local   writes an adf34 compatible configuration.
;            adas808_cfg_cmplx ADAS    return configurations of an n-shell complex
;            h8nlev            ADAS    returns the level count for a configuration
;
;
; CATEGORY:
;       Adas system.
;
; WRITTEN:
;       H. P. Summers, University of Strathclyde,29-06-06.
;
; MODIFIED:
;       1.1     H. P. Summers
;                 - First version put into CVS.
;       1.2     Allan Whiteford
;                 - Changed call from complexes to adas808_cfg_complx
;     Was missed during renaming exercise.
;       1.3     H. P. Summers
;                 - Added lonarr for config,term and level count vector; improved
;                   input/output descriptors.  Correction to logic for rare gas
;                   omitted closed-shell detection
;       1.4     H. P. Summers
;                 - Further correction to logic for Cowan effective z_ion for adf34
;                   driver.  Now use zc_cow=z_ion for z0_nuc<19.
;       1.5     H. P. Summers
;                 - Correction to preamble text for fill_par
```

```
;       1.6     H. P. Summers
;                 - Replaced variable name z0 by z0_nuc, z1 by z_ion and zc by zc_cow to
;                   avoid confusion.  Introduced rules structure as a keyword parameter
;
; VERSION:
;       1.1   25-08-06
;       1.2   09-10-06
;       1.3   29-11-06
;       1.4   18-06-07
;       1.5   18-07-08
;       1.6   22-08-08
;
;-
;-----------------------------------------------------------------------------
PRO adas8xx_promotions,  z0_nuc     = z0_nuc,                       $
                         z_ion      = z_ion,                        $
                         ionpot     = ionpot,                       $
                         rules      = rules,                        $
 grd_cfg    = grd_cfg,                          $
                         grd_occ    = grd_occ,                      $
                         ex_cfg     = ex_cfg,                       $
                         grd_par    = grd_par,                      $
                         ex_par     = ex_par,                       $
                         grd_zc_cow     = grd_zc_cow,               $
                         ex_zc_cow      = ex_zc_cow,                $
                         oc_store   = oc_store,                     $
 no_configs = no_configs,                       $
 no_terms   = no_terms,                         $
 no_levels  = no_levels

;-----------------------------------------------------------------------------
```

Notes:

cfg2occ.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  cfg2occ
;
; PURPOSE    :  Converts a textual configuration to an occupation
;               number array
;
; EXPLANATION:
;       This function takes a textual configuration, nuclear charge
; and ion charge to return occupation numbers of electrons
;
; USE:
;       An example;
;               result = cfg2occ('2s 2p',6,2)
; print,result
```

```
; 2 1 1 0 0 0 0 0 0 0 0 0 0 ...
;
; INPUTS:
;       CFG: The configuration in either standard or Eissner
;            notation.
;       IZ0: The nuclear charge of the ion.
;       ZZ : The charge of the ion.
;
; OPTIONAL INPUTS:
;       None.
;
; OUTPUTS:
;       An occupation number array with all electrons specified
;
; OPTIONAL OUTPUTS:
;       None.
;
; KEYWORD PARAMETERS:
;       help : Displays documentation
;
; CALLS:
;       read_adf00 for ground configurations.
;
; SIDE EFFECTS:
;       None
;
; NOTES:
;       Determines unspecified electrons by filling in occupation
;       numbers based on the ground configuration of the ion in
;       question in order of spectroscopic notation, this means for
;       the W+14 with a configuration of simply '5p' it will assume
;       that a 5s electron has been promoted rather than a 4f.
;
;       Note, however, that the routine does not assume lower
;       occupation numbers are filled if the ground configuration
;       has them partially empty so for neutral tungsten with
;       a configuration of '6p' it will leave the 5f and 5g orbitals
;       empty and the 5d partially filled as one would expect.
;
; AUTHOR     :  Allan Whiteford
;
; DATE       :  12-10-04
;
;
; MODIFIED:
;       1.1     Allan Whiteford
;               - First version.
;
; VERSION:
;       1.1    31-07-08
;-
;-----------------------------------------------------------------------

function cfg2occ,cfg,iz0,iz,help=help

;-----------------------------------------------------------------------------
```

Notes:

adas8xx_create_drivers.pro

```
;-----------------------------------------------------------------------------
;+
;
; NAME     : adas8xx_create_drivers
;
; PURPOSE  : prepares the driver data sets of format adf34 and adf42 for
;            complete heavy element structure and emissivity calculations.
;            Default names are setup if keywords are omitted on call.
;
; CATEGORY : ADAS
;
; NOTES    : adf04, adf15 and adf40 data set names are required.  Default
;            names are setup if keywords are omitted on call.
;
;            The 'plasma' structure is defined as
;
;              theta[]       : full vector of temperatures (eV) - unscaled
;              indx_theta[]  : pointer vector to temperature values in theta[] to be used.
;              rho[]      : full vector of densities (cm-3) - unscaled
;              indx_rho[]    : pointer vector to density values in rho[] to be used.
;              npix[]      : full vector of pixel counts of wavelength ranges
;              wvlmin[]      : full vector of lower wavelength limit (Angstrom) of wavelength ranges
;              wvlmax[]        full vector of upper wavelength limit (Angstrom) of wavelength ranges
;              indx_wvl[]    : pointer vector to wavelength ranges in npix[],wvlmin[] and
;                     : wvlmax[] to be used.
;
;
;            The 'files' structure is defined as
;
;              adf34_file        : adf34 file name
;              adf34_inst_file   : adf34 inst file name
;              adf34_ls_pp_file  : adf34 pp file name for 'ls' resolution
;              adf34_ic_pp_file  : adf34 pp file name for 'ic' resolution
;              adf42_ca_file     : adf42 file name for 'ca' resolution
;              adf42_ls_file     : adf42 file name for 'ls' resolution
;              adf42_ic_file     : adf42 file name for 'ic' resolution
;        adf42_ca_pp_file  : adf42 pp file name for 'ca' resolution
;              adf42_ls_pp_file  : adf42 pp file name for 'ls' resolution
;              adf42_ic_pp_file  : adf42 pp file name for 'ic' resolution
;        adf04_ca_t1_file  : adf04 - type 1 file name for 'ca' resolution
;        adf04_ca_file     : adf04 file name for 'ca' resolution
;        adf04_ls_file     : adf04 file name for 'ls' resolution
;        adf04_ic_file     : adf04 file name for 'ic' resolution
;        adf15_ca_file  : adf15 file name for 'ca' resolution
;        adf15_ls_file  : adf15 file name for 'ls' resolution
;        adf15_ic_file  : adf15 file name for 'ic' resolution
;        adf40_ca_file  : adf40 file name for 'ca' resolution
;        adf40_ls_file  : adf40 file name for 'ls' resolution
;        adf40_ic_file  : adf40 file name for 'ic' resolution
;        adf11_ca_file  : adf11 file name for 'ca' resolution
;        adf11_ls_file  : adf11 file name for 'ls' resolution
```

```
;        adf11_ic_file  : adf11 file name for 'ic' resolution
;
;
; USE      : Usually called after 'adas8xx_promotion_rules'
;
; INPUT    :
;
;   (I*4)   z0              = keyword = nuclear charge
;   (I*4)   z1              = keyword = selected ion charge
;   (C* )   config          = keyword = ground configuration for selected ion
;   (R*8)   ionpot          = keyword = ionis. potential (ev) for selected ion
;   (R*8)   theta[]         = keyword = full vector of temperatures (eV) - unscaled
;   (I*4)   indx_theta[]    = keyword = pointer vector to temperature values in theta[] to be used.
;   (R*8)   rho[]           = keyword = full vector of densities (cm-3) - unscaled
;   (I*4)   indx_rho[]      = keyword = pointer vector to density values in rho[] to be used.
;   (I*4)   npix[]          = keyword = full vector of pixel counts of wavelength ranges
;   (R*8)   wvlmin[]        = keyword = full vector of lower wavelength limit (Angstrom) of wavelengt
;   (R*8)   wvlmax[]        = keyword = full vector of upper wavelength limit (Angstrom) of wavelengt
;   (I*4)   indx_wvl[]      = keyword = pointer vector to wavelength ranges in npix[],wvlmin[] and wv
;
;   (str)   plasma          = keyword = structure containing theta, indx_theta, rho, indx_rho,
;                                                            npix, wvlmin, wvlmax, indx_wvl
;
;   (C* )   adf34_file      = keyword = adf34 file name
;   (C* )   adf34_inst_file = keyword = adf34 inst file name
;   (C* )   adf34_ls_pp_file = keyword = adf34 pp file name for 'ls' resolution
;   (C* )   adf34_ic_pp_file = keyword = adf34 pp file name for 'ic' resolution
;   (C* )   adf42_ls_file   = keyword = adf42 file name for 'ls' resolution
;   (C* )   adf42_ic_file   = keyword = adf42 file name for 'ic' resolution
;   (C* )   adf42_ca_file   = keyword = adf42 file name for 'ca' resolution
;   (C* )   adf42_ls_pp_file = keyword = adf42 pp file name for 'ls' resolution
;   (C* )   adf42_ic_pp_file = keyword = adf42 pp file name for 'ic' resolution
;   (C* )   adf42_ca_pp_file = keyword = adf42 pp file name for 'ca' resolution
;   (C* )   adf04_ca_t1_file = keyword = adf04 - type 1 file name for 'ca' resolution
;   (C* )   adf04_ca_file   = keyword = adf04 file name for 'ca' resolution
;   (C* )   adf04_ls_file   = keyword = adf04 file name for 'ls' resolution
;   (C* )   adf04_ic_file   = keyword = adf04 file name for 'ic' resolution
;   (C* )   adf40_ca_file   = keyword = adf40 file name for 'ca' resolution
;   (C* )   adf40_ls_file   = keyword = adf40 file name for 'ls' resolution
;   (C* )   adf40_ic_file   = keyword = adf40 file name for 'ic' resolution
;   (C* )   adf11_ca_file   = keyword = adf11 file name for 'ca' resolution
;   (C* )   adf11_ls_file   = keyword = adf11 file name for 'ls' resolution
;   (C* )   adf11_ic_file   = keyword = adf11 file name for 'ic' resolution
;
;   (str)   files           = keyword = structure containing adf34_file, adf34_inst_file, adf34_ls_pp
;                                                            adf34_ic_pp_file, adf42_ca_file, adf42_l
;                                                            adf42_ic_file, adf42_ca_pp_file, adf42_l
;                                                            adf42_ic_pp_file, adf04_ca_t1_file, adf0
;                                                            adf04_ls_file, adf04_ic_file, adf40_ca_f
;                                                            adf40_ls_file, adf40_ic_file, adf11_ca_f
;                                                            adf11_ls_file, adf11_ic_file
;
;   (I*4)   for_tr_sel      = keyword = Cowan option for radiative transitions 1 - first parity, 2 or
;   (C* )   grd_cfg         = keyword = ground configuration
;   (C* )   ex_cfg[]        = keyword = excited configurations
;   (I*4)   grd_par         = keyword = ground parity
;   (I*4)   ex_par[]        = keyword = excited parities
```

```
;   (I*4)   grd_zc           = keyword = grd. eff. charge for Cowan adf34 driver
;                                        n.b. prescription ok for z0>19.  For
;                                             z0<19 revert to zc=z1=ion charge+1
;   (R*8)   ex_zc[]          = keyword = exc. eff. charge for Cowan adf34 driver
;                                        n.b. prescription ok for z0>19.  For
;                                             z0<19 revert to zc=z1=ion charge+1
;   (I*4)   /ca_only         = keyword => prepare only config. average drivers
;   (I*4)   /make_ic_adf34   = keyword => force preparation of 'ls' and 'ic' adf34 drivers even if /ca
;
; OUTPUTS   :
;
;
; ROUTINES:
;            NAME              TYPE    COMMENT
;            xxdate            system  returns the date
;            xxesym            ADAS    returns element symbol for given nuclear charge
;
;
; CATEGORY:
;       Adas system.
;
; WRITTEN:
;       H. P. Summers, University of Strathclyde,29-06-06.
;
; MODIFIED:
;       1.1     H. P. Summers
;               First version put into CVS.
;       1.2     H. P. Summers
;               Update comments and keyword checks/defaults
;       1.3     H. P. Summers
;               Added structures 'plasma' and 'files', their default handling and their
;               relationship to their separate keyword parameters
;
; VERSION:
;       1.1   25-08-06
;       1.2   06-08-08
;       1.3   21-08-08
;
;-
;-----------------------------------------------------------------------------
PRO adas8xx_create_drivers, z0              = z0,                           $
                            z1              = z1,                           $
                            zc              = zc,                           $
                            ionpot          = ionpot,                       $
                            theta           = theta,                        $
                            indx_theta      = indx_theta,                   $
                            rho             = rho,                          $
                            indx_rho        = indx_rho,                     $
                            npix            = npix,                         $
                            wvlmin          = wvlmin,                       $
                            wvlmax          = wvlmax,                       $
                            indx_wvl        = indx_wvl,                     $
    plasma          = plasma,                       $
                            adf34_file      = adf34_file,                   $
                            adf34_inst_file = adf34_inst_file,              $
                            adf34_ls_pp_file = adf34_ls_pp_file,            $
                            adf34_ic_pp_file = adf34_ic_pp_file,            $
```

```
                          adf42_ca_file    = adf42_ls_file,                $
                          adf42_ls_file    = adf42_ic_file,                $
                          adf42_ic_file    = adf42_ca_file,                $
                          adf42_ca_pp_file = adf42_ls_pp_file,             $
                          adf42_ls_pp_file = adf42_ic_pp_file,             $
                          adf42_ic_pp_file = adf42_ca_pp_file,             $
                          adf04_ca_t1_file = adf04_ls_file,                $
                          adf04_ca_file    = adf04_ic_file,                $
    adf04_ls_file    = adf04_ls_file,                $
    adf04_ic_file    = adf04_ic_file,                $
    adf15_ca_file    = adf15_ca_file,                $
    adf15_ls_file    = adf15_ls_file,                $
    adf15_ic_file    = adf15_ic_file,                $
    adf40_ca_file    = adf40_ca_file,                $
    adf40_ls_file    = adf40_ls_file,                $
    adf40_ic_file    = adf40_ic_file,                $
    adf11_ca_file    = adf11_ca_file,                $
    adf11_ls_file    = adf11_ls_file,                $
    adf11_ic_file    = adf11_ic_file,                $
    files            = files,                        $
                          for_tr_sel       = for_tr_sel,                   $
                          grd_cfg          = grd_cfg,                      $
                          ex_cfg           = ex_cfg,                       $
                          grd_par          = grd_par,                      $
                          ex_par           = ex_par,                       $
                          grd_zc           = grd_zc,                       $
                          ex_zc            = ex_zc,                        $
                          ca_only          = ca_only,                      $
                          make_ic_adf34    = make_ic_adf34


;-------------------------------------------------------------------------------
```

Notes:

adas8xx_create_ca_adf04.pro

```
;-------------------------------------------------------------------------------
;+
; PROJECT : ADAS
;
; NAME    : adas8xx_create_ca_adf04
;
; PURPOSE : Generates configuration average adf04 files
;
; EXPLANATION:
;       This routines takes a given set of configurations, nuclear and ionic
;       charge and generates a complete adf04 file (type I and/or type III)
;       working in a PWB CA approximation.
;
; USE:
;       An example;
;
```

```
;       iz0=6
;       iz=3
;
;       occup=[ [2,1,0,0,0,0], $
;               [2,0,0,0,0,1], $
;               [2,0,0,1,0,0], $
;               [2,0,0,0,1,0], $
;               [2,0,1,0,0,0] $
;               ]
;
;       caex,iz,iz0,occup,adf04_t1_file='adf04_1',adf04_t3_file='adf04_3'
;
; INPUTS:
;       iz :   Ion charge
;       iz0:   Nuclear charge
;       occup: Occupation numbers of configurations
;                    First index:  configuration number
;                    Second index: orbital number.
;
; OPTIONAL INPUTS:
;       None.
;
; OUTPUTS:
;       None.
;
; OPTIONAL OUTPUTS:
;       exit_status: returns -1 if ADF04 generation has failed for any reason
;
; KEYWORD PARAMETERS:
;       ionpot:    Ionisation potential, if not specified it will be read from central ADAS
;       adf04_t1_file: Filename of type I file to produce
;       adf04_t3_file: Filename of type III file to produce
;       keeppass:  Do not remove temporary passing files.
;       occ2cow:   Convert occupation numbers to Cowan (i.e. adas801) input standard
;
; CALLS:
;       read_adf00: read adf00 files.
;       h4mxwl: Maxwell average collision strengths.
;       xxspln: Used for splining (momentum transfer)
;
; SIDE EFFECTS:
;       This function spawns UNIX commands.
;
; CATEGORY:
;       Series 8 utility.
;
; WRITTEN:
;       Allan Whiteford
;
; MODIFIED:
;       Version 1.1   Allan Whiteford    01-11-2004
;                     First release (originally h8caex).
;       Version 1.2   Hugh Summers       28-09-2006
;                     Major revision and name change to
;                     adas8xx_create_ca_adf04
;       Version 1.3   Hugh Summers       04-12-2006
;                     Added call to adas8xx_cowan_string_check
```

```
;       Version 1.4   Hugh Summers       24-07-2007
;                     Modified temporary filenames to make unique
;       Version 1.5   Adam Foster        08-01-2008
;                     Modified to return 'exit_status'. Also quits
;                     rather than crashes if Cowan run has failed
;       Version 1.6   Hugh Summers       24-07-2007
;                     Put ,/sh on spawning final copying to archive

; VERSION:
;       1.1     01-11-2004
;       1.2     28-09-2006
;       1.3     04-12-2006
;       1.4     24-07-2007
;       1.5     08-01-2008
;       1.6     25-07-2008
;-
;-----------------------------------------------------------------
pro adas8xx_create_ca_adf04, iz,                                  $
                             iz0,          $
                             occup,          $
                             ionpot      = ionpot,           $
                             theta      = theta,         $
                             indx_theta      = indx_theta,       $
                             adf04_t1_file   = adf04_t1_file,    $
                             adf04_t3_file   = adf04_t3_file,    $
                             archive_dir     = archive_dir,      $
                             archive_files   = archive_files,    $
                             keeppass      = keeppass,          $
                             exit_status     = exit_status

;-----------------------------------------------------------------------------
```

Notes:

adas8xx_create_ls_ic_adf04.pro

```
;-----------------------------------------------------------------------------
PRO adas8xx_create_ls_ic_adf04, z0,                                $
                                z1,                                $
                                adf34_file        = adf34_file,        $
                                adf34_inst_file   = adf34_inst_file,   $
                                adf34_ls_pp_file  = adf34_ls_pp_file,  $
                                adf34_ic_pp_file  = adf34_ic_pp_file,  $
                                adf04_ls_file     = adf04_ls_file,     $
                                adf04_ic_file     = adf04_ic_file,     $
                                archive_dir       = archive_dir,       $
                                archive_files     = archive_files,     $
                                submit            = submit


;-----------------------------------------------------------------------------
```

Notes:

run_adas808.pro

```
;-----------------------------------------------------------------------------
;+
;
; NAME     : run_adas808
;
; PURPOSE  : generate configuration sets for the ions of elements and
;            write all ls-, ic- and ca-coupling adf34 and adf42 driver and
;            data files.  Execute configuration average Cowan structure
;            calculation and write ca- type 1 and type 3 adf04 files.
;            Create ls- and ic- adf04 files.  Create  ls- ic- and ca-
;            adf15 and adf40 and adf11 (partial plt) files.
;
; CATEGORY : ADAS
;
; NOTES    :
;
; USE      : This is a set up procedure for both offline and online use of
;            adas801, adas808, adas810.  It is designed for economised
;            handling of heavy species and subsequent use in a partitioned
;            superstage framework.
;
;            Use of consistent temperature, density, wavelength ranges
;            throughout large scale production is achieved by inclusion
;            of defaults sets in the procedure with working subsets
;            specified by indexing vectors
;
;                 indx_theta: pointing to the temperature vector theta
;                 indx_rho  : pointing to the density vector rho
;                 indx_wvl  : pointing to the npix, wvlmin, wvlmax vectors
;
;            The default ADAS subsets are adopted by omitting the keywords
;            in calling the procedure.
;
;            By default, the temperatures theta[] are treated as reduced
;            to be scaled with the ion charge, z, as te[]=(z+1)^2*theta[],
;            and are treated as absolute by using the keyword /theta_noscale
;
;            By default the densities rho[] are treated as absolute are
;            scaled only if the keyword /rho_scale is used.  Then
;            dens[]=rho[]/(z+1)^7.
;
;            The internal settings are:

;            theta = [ 2.00e+02, 3.00e+02, 5.00e+02, 7.00e+02, 1.00e+03,
;                      1.50e+03, 2.00e+03, 3.00e+03, 5.00e+03, 7.00e+03,
;                      1.00e+04, 1.50e+04, 2.00e+04, 3.00e+04, 5.00e+04,
;                      7.00e+04, 1.00e+05, 1.50e+05, 2.00e+05, 3.00e+05,
;                      5.00e+05, 1.00e+06, 2.00e+06, 5.00e+06, 1.00e+07]
;
;            indx_theta = [ 0, 2, 4, 6, 8,10,12,14,16,18,20,21,22,23]
;
```

```
;            rho   = [ 1.00e-03, 1.00e+00, 1.00e+01, 1.00e+02, 1.00e+03,
;                     1.00e+04, 3.00e+04, 1.00e+05, 3.00e+05, 1.00e+06,
;                     3.00e+06, 1.00e+07, 3.00e+07, 1.00e+08, 3.00e+08,
;                     1.00e+09, 3.00e+09, 1.00e+10, 3.00e+10, 1.00e+11,
;                     3.00e+11, 1.00e+12, 3.00e+12, 1.00e+13, 3.00e+13,
;                     1.00e+14, 3.00e+14, 1.00e+15, 3.00e+15, 1.00e+16]
;
;            indx_rho   = [17,19,21,23,25,27,29]
;
;
;            npix  = [      128,      128,      128,      128,      512,
;                           256,      256,      256,      256,      256]
;
;            wvlmin= [ 1.00e+00, 1.00e+01, 1.00e+02, 1.00e+03, 1.00e+00,
;                     3.00e+00, 7.00e+01, 5.00e+02, 1.00e+03, 3.00e+03]
;
;            wvlmax= [ 1.00e+01, 1.00e+02, 1.00e+03, 1.00e+04, 1.00e+04,
;                     5.00e+00, 1.20e+02, 1.50e+03, 2.00e+03, 7.00e+03]
;
;            indx_wvl   = [ 0, 1, 4]
;
; INPUT    :
;
;   (I*4)   z0_list        = set of nuclear charge of required elements
;   (I*4)   nel_min        = lowest number of bound electrons for ion range
;   (I*4)   nel_max        = largest number of bound electrons for ion range
;   (I*4)   indx_theta[]   = keyword = pointer vector to temperature
;                                      values  in theta[] to be used.
;   (I*4)   indx_rho[]     = keyword = pointer vector to density
;                                      values  in rho[] to be used.
;   (I*4)   indx_wvl[]     = keyword = pointer vector to wavelength
;                                      ranges in npix[],wvlmin[] and
;                                      wvlmax[] to be used.
;   (C  )   a54file        = keyword => adf54 file name (full pathway)

;   (I*4)   /theta_noscale = keyword => treat theta[] as absolute
;   (I*4)   /rho_scale     = keyword => treat rho[] as reduced
;   (I*4)   /ca_only       = keyword => compute only config. average results
;
;
; ROUTINES:
;          NAME                            TYPE   COMMENT
;          xxelem                          adas   obtain an element name
;          xxesym                          adas   returns an element chemical symbol
;          adas8xx_promotion_rules         adas   set promotion rules from adf54 file
;          adas8xx_promotions              adas   create configuration set for an ion
;          adas8xx_create_drivers          adas   write adf34/adf42 drivers for an ion
;          adas8xx_create_ca_adf04         adas   create 'ca' adf04 type 1 and 3 files
;          adas8xx_create_ls_ic_adf04      adas   create 'ls' and 'ic' adf04 type 1
;                                                 and 3 files
;          adas8xx_create_adf15_adf40      adas   create 'ca', 'ls' and 'ic' adf15,
;                                                 adf40 and adf11(partial plt) files
;
;
; CATEGORY:
;       Adas system.
;
```

```
; WRITTEN:
;       H. P. Summers, University of Strathclyde,22-08-06.
;
; MODIFIED:
;       1.1     H. P. Summers
;               First version put into CVS.
;       1.2     H. P. Summers
;               Added ca_only and a54file keywords and associated
;               'small', 'medium' and 'large' default promotion rules.
;               Adjusted to long integers for term/level counts and
;               tidied count summary print out.
;       1.3     H. P. Summers
;               Correction to z0_list when nel_min gt z0.
;       1.4     A. Foster & H. P. Summers
;               Changed to adf54 data set for promotion rules and
;               returned to a single adas8xx_promotions_rule.pro
;               which fetches in the adf54.
;
; VERSION:
;       1.1   25-08-06
;       1.2   17-01-07
;       1.3   05-07-07
;       1.4   23-07-08
;
;-
;-----------------------------------------------------------------------------

PRO run_adas808,                z0_list       = z0_list,              $
                                nel_min       = nel_min,              $
                                nel_max       = nel_max,              $
                                indx_theta    = indx_theta,           $
                                indx_rho      = indx_rho,             $
                                indx_wvl      = indx_wvl,             $
                                a54file       = a54file,              $
                                theta_noscale = theta_noscale,        $
                                rho_scale     = rho_scale,            $
                                ca_only       = ca_only


common shells, ndshell     ,                                        $
               shmax       , nval        , lval       , shl_lab  ,  $
               iprom_1     , iprom_2     ,                          $
               nrg_cnt_idx , nrg_exl_idx , nrg_el_cnt ,              $
               nval_max    , nval_min    , lval_max   , lval_min ,  $
               njvals

;-----------------------------------------------------------------------------
```

Notes:

adas8xx_opt_promotions_control.pro

```
;-----------------------------------------------------------------------------
```

Notes:

adas8xx_opt_expand_promotions.pro

```
;-----------------------------------------------------------------------------
```

Notes:

r8fbch.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  r8fbch
;
; PURPOSE    :  Calculates a shell contribution to the ionisation rate
;               coefficient in the Burgess-Chidichimo approximation.
;
; ARGUMENTS  :  All output arguments will be defined appropriately.
;               Inputs will be converted to correct type, if possible,
;               internally without changing calling type.
;                   result = r8fbch(iz=iz, xi=xi, zeta=zeta, te=te)
;
;               r8fbch, iz     = iz,    $
;                       xi     = xi,    $
;                       zeta   = zeta,  $
;                       te     = te
;
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  iz         I     integer recombined ion charge
;               xi         I     double  effective ionisation potential (Ryd)
;               zeta       I     double  effective number of equivalent electrons
;               te         I     double  array of electron temperature (K)
;
;
; KEYWORDS      help       I      -      prints help to screen
;
;
; NOTES      :  Calls the fortran code.
;               Units of result are cm^3/s
;
;
; AUTHOR     :  Martin O'Mullane
;
; DATE       :  18-09-2008
```

```
;
;
; MODIFIED:
;         1.1       Martin O'Mullane
;                       - First version.
;
; VERSION:
;         1.1      18-09-2008
;-
;-----------------------------------------------------------------------

FUNCTION r8fbch, iz     = iz,    $
                 xi     = xi,    $
                 zeta   = zeta,  $
                 te     = te,    $
                 help   = help

;-------------------------------------------------------------------------------
```

Notes:

r8necip.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  r8necip
;
; PURPOSE    :  Calculates ECIP approximation for ionisation rate.
;
; ARGUMENTS  :  All output arguments will be defined appropriately.
;               Inputs will be converted to correct type, if possible,
;               internally without changing calling type.
;                   result = rsnecip(iz=iz, xi=xi, zeta=zeta)
;
;               r8necip, iz     = iz,    $
;                        xi     = xi,    $
;                        zeta   = zeta,  $
;                        te     = te,    $
;                        alfred = alfred
;
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  iz         I     integer recombined ion charge
;               xi         I     double  effective ionisation potential (Ryd)
;               zeta       I     double  effective number of equivalent electrons
;               te         I     double  array of electron temperature (K)
; OPTIONAL      alfred     O     double  scaled 3-body recombination coefficient
;
;
; KEYWORDS      None
;
```

```
;
; NOTES      :  Calls the fortran code.
;               Units of result are cm^3/s
;
;
; AUTHOR     :  Martin O'Mullane
;
; DATE       :  08-04-2002
;
;
; MODIFIED:
;         1.1       Martin O'Mullane
;                       - First version.
;   1.2     Richard Martin
; - Removed underscore in CALL_EXTERNAL statement.
;   1.3     Allan Whiteford
; - Changed wrapper path to be just ADASFORT.
;
; VERSION:
;         1.1      08-04-2002
;   1.2    17-03-2003
;   1.3    10-08-2004
;-
;-----------------------------------------------------------------------

FUNCTION r8necip, iz     = iz,    $
                  xi     = xi,    $
                  zeta   = zeta,  $
                  te     = te,    $
                  alfred = alfred

;-------------------------------------------------------------------------------
```

Notes:

config_orbital_energies.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  config_orbital_energies
;
; PURPOSE    :  Calculates the shell orbital energies for the ground
;               configuration, or supplied conffiguration, of an ion.
;
; EXPLANATION:
;               Configurations are taken from adf00 datasets if none are
;               supplied. RCN from Cowan is run and orbital energies, along
;               with nlq are returned.
;
;
; ARGUMENTS  :  All output arguments will be defined appropriately.
```

```
;
;               NAME      I/O    TYPE   DETAILS
; REQUIRED   :  z0_nuc     I     int    atomic number
;               z_ion      I     int    ionisation stage
; OPTIONAL      config     I     str    full configuration string suitable for adf34
;               n          O     int()  principal quantum number
;               l          O     int()  angular quantum number
;               q          O     int()  occupation number
;               energy     O     real() orbital energy for nlq shell
;               elec       O     int    number of electrons from RCN output file
;
; KEYWORDS
;               help       I      -     prints help to screen
;
; NOTES      :   - Spawns rcn.x and get_orbital.x. Can leave temporary files in
;                  working directory if exits uncleanly.
;                - The configuration string must start ar 1s and list all shells.
;                  It must be suitable for inclusion as adf34. See any adf00
;                  configuration for a suitable example.
;
; AUTHOR     :  Martin O'Mullane
;
; DATE       :  08-10-2008
;
;
; MODIFIED:
;       1.1     Martin O'Mullane
;               - First version.
;
; VERSION:
;       1.1    08-10-2008
;-
;-----------------------------------------------------------------------------

PRO config_orbital_energies, z0_nuc  = z0_nuc,  $
                             z_ion   = z_ion,   $
                             config  = config,  $
                             n       = n,       $
                             l       = l,       $
                             q       = q,       $
                             energy  = energy,  $
                             elec    = elec,    $
                             help    = help

;-----------------------------------------------------------------------------
```

Notes:

tev_alf_s.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT:
```

```
;       ADAS
;
; NAME:
;       tev_alf_s
;
; PURPOSE:
;       Returns the electron temperature in eV at which the
;       recombination coefficient alf(z+1 --> z) equals the
;       ionisation coefficient S(z --> z+1) using Seaton (1964)
;       expressions for the coefficients.
;
; ARGUMENTS  :  Arguments are non-positional named parameters.   All output
;               arguments will be defined appropriately.
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED:     z_ion      I     real  ionising ion charge (= recombined ion charge).
;               ionpot_z   I     real  ionisation potential of ion z (Ryd).
;               tev_alf_s  O     real  function returned value = elect. temperature (K).
; OPTIONAL:     zeta       I     real  number of equiv. electrons for ionis. coeff.
:                                        (defaults to 1.0).
;               ph_frac    I     real  phase space avail. frac. for recom. coeff.
;                                        (defaults to 1.0).
;               accur      I     real  frac. change in temperature at which
;                                          iteration terminates (defaults to 0.001).
;
;
; KEYWORDS   :  ev         I      -      switch ionpot_z and tev_alf_s to elec. volts
;               help       I      -      prints help to screen
;
; NOTES      :
;
;
; AUTHOR:   Hugh Summers
:
; DATE:     9-10-2008
;
; MODIFIED:
;       1.1     Hugh Summers
;                - First release
;
; VERSION:
;       1.1     9-10-2008
;
;-----------------------------------------------------------------------------

FUNCTION tev_alf_s, z_ion      = z_ion,      $
                    ionpot_z   = ionpot_z,   $
    zeta       = zeta,       $
    ph_frac    = ph_frac,    $
    accur      = accur,      $
    ev         = ev,         $
    help       = help


;-----------------------------------------------------------------------------
```

Notes:

sbchid_cfg_tot.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  sbchid_cfg_tot
;
; PURPOSE    :  calculates total ionisation rate coefficient for a
;               configuration from IDL, using  the Burgess-Chidichimo
;               expression for shell contributions.
;
; ARGUMENTS  :  Arguments are non-positional named parameters.   All output
;               arguments will be defined appropriately.
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  z0_nuc     I     int     nuclear charge of element
;               z_ion      I     int     charge of ionising ion
;               te         I     real()  array of electron temperatures (K)
;               coef       O     real()  array of ionisation coefficients (cm^3 s^-1)
; OPTIONAL   :  config_z   I     str     full configuration string of initial state
:                                          (defaults to adf00)
;               config_z1  I     str     full configuration string of final state
;                                          (defaults to adf00)
;               ionpot_z   I     real    ionisation potential of initial state to ionised
;                                          ion ground(Ryd)(defaults to adf00)
;               excpot_z1  I     real    excitation energy of final state above ionised
;                                          ion ground (Ryd)(defaults to zero)
;               case_ionis I     str     selected approximation case (‘case a’,
;                                          ’case ba’ etc).(defaults to ’case bb’).
;               ci_v       I     real    scaling factor for valence ionisation group
;                                          (defaults to 1.0)
;               ci_nv      I     real    scaling factor for non-valence ionisation group
;                                          (defaults to 1.0)
;               cr         I     real    scaling factor for excitation group
;                                          (defaults to 1.0)
;
; KEYWORDS   :  ev         I      -      switch ionpot_z and te[] to elec. volts
;               help       I      -      prints help to screen
;
; NOTES      :
;
;
; AUTHOR     :  Hugh Summers
;
; DATE       :  28-11-2008
;
;
; MODIFIED:
;       1.1     Hugh Summers
;                 - first release
;
; VERSION:
```

```
;       1.1     28-11-2008
;
;-
;-------------------------------------------------------------------------

PRO sbchid_cfg_tot, z0_nuc     =  z0_nuc,     $
                    z_ion      =  z_ion,      $
                    config_z   =  config_z,   $
                    config_z1  =  config_z1,  $
                    ionpot_z   =  ionpot_z,   $
                    excpot_z1  =  excpot_z1,  $
    case_ionis =  case_ionis, $
    ci_v       =  ci_v,       $
    ci_nv      =  ci_nv,      $
    ci_r       =  ci_r,       $
                      te         =  te,      $
                      coef       =  coef,       $
    ev         =  ev,         $
                      help       =  help

;-------------------------------------------------------------------------
```

Notes:

read_adf56.pro

```
;-----------------------------------------------------------------------------
```

Notes:

adas8xx_ionis_promotion_rules.pro

```
;-----------------------------------------------------------------------------
```

Notes:

adas8xx_ionis_promotions.pro

```
;-----------------------------------------------------------------------------
```

Notes:

adas8xx_ionis_create_drivers.pro

```
;-----------------------------------------------------------------------------
```

Notes:

adas8xx_ionis_create_ca_adf23.pro

```
;-----------------------------------------------------------------------------
```

Notes:

run_adas813.pro

```
;-----------------------------------------------------------------------------
```

Notes:

alf_r_bdn.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  alf_r_bdn
;
; PURPOSE    :  calculates radiative recombination coefficients to an
;               n-shell at a set of electron temperatures.
;
; ARGUMENTS  :  Arguments are non-positional named parameters.   All output
;               arguments will be defined appropriately.
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  z0_nuc     I     int     nuclear charge of element
;               z_ion      I     int     charge of recombined ion
;               n          I     int     principal quantum number of final state
;               te         I     real()  array of electron temperatures (K)
;               coef       O     real()  array of radiative recom. coeffts (cm^3 s^-1)
; OPTIONAL   :  ionpot_n   I     real    ionisation potential for final state
;                                        n-shell to recombining ion ground(Ryd)
;                                           (defaults to hydrogenic energy)
;               approx     I     str     selected approximation (‘no-gf’,’h-gf’).
;                                           (defaults to ’h-gf’).
;
```

```
; KEYWORDS   :  ev         I      -      switch ionpot_z and te[] to elec. volts
;               help       I      -      prints help to screen
;
; NOTES      :
;
;
; AUTHOR     :  Hugh Summers
;
; DATE       :  28-11-2008
;
;
; MODIFIED:
;       1.1     Hugh Summers
;                 - first release
;
; VERSION:
;       1.1     28-11-2008
;
;-
;-----------------------------------------------------------------------

PRO alf_r_bdn, z0_nuc     =  z0_nuc,     $
               z_ion   =  z_ion, $
               n   =  n,   $
               te   =  te, $
               coef   =  coef, $
               ionpot_n   =  ionpot_n,   $
               approx   =  approx, $
               ev   =  ev,   $
               help   =  help

;-------------------------------------------------------------------------------
```

Notes:

alf_r_bdnl.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  alf_r_bdnl
;
; PURPOSE    :  calculates radiative recombination coefficients to an
;               nl-shell at a set of electron temperatures.
;
; ARGUMENTS  :  Arguments are non-positional named parameters.   All output
;               arguments will be defined appropriately.
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  z0_nuc     I     int     nuclear charge of element
;               z_ion      I     int     charge of recombined ion
;               n          I     int     principal quantum number of final state
```

```
;               l          I     int      orbital quantum number of final state
;               te         I     real()   array of electron temperatures (K)
;               coef       O     real()   array of radiative recom. coeffts (cm^3 s^-1)
; OPTIONAL   :  config_z1  I     str      full configuration string of initial state
;                                           (defaults to adf00)
;               ionpot_nl  I     real     ionisation potential for final nl-shell
;                                         to recombining ion initial state(Ryd)
;                                           (defaults to hydrogenic energy)
;               approx     I     str      selected approximation (‘no-gf’,’h-gf’,
;                                           ’dw-gf’).(defaults to ’h-gf’).
;
; KEYWORDS   :  ev         I      -       switch ionpot_z and te[] to elec. volts
;               help       I      -       prints help to screen
;
; NOTES      :
;
;
; AUTHOR     :  Hugh Summers
;
; DATE       :  28-11-2008
;
;
; MODIFIED:
;       1.1     Hugh Summers
;                 - first release
;
; VERSION:
;       1.1     28-11-2008
;
;-
;-----------------------------------------------------------------------

PRO alf_r_bdnl, z0_nuc     =  z0_nuc,     $
                z_ion    =  z_ion,  $
                n    =  n,  $
                l    =  l,  $
                te    =  te,  $
                coef    =  coef,  $
                config_z1  =  config_z1,  $
                ionpot_n   =  ionpot_n,   $
                approx     =  approx,  $
                ev    =  ev,  $
                help    =  help

;-----------------------------------------------------------------------------
```

Notes:

alf_r_tot.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
```

```
;
; NAME       :  alf_r_tot
;
; PURPOSE    :  calculates total radiative recombination coefficients to an
;               ion at a set of electron temperatures and electron densities.
;
; ARGUMENTS  :  Arguments are non-positional named parameters.   All output
;               arguments will be defined appropriately.
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  z0_nuc     I     int     nuclear charge of element
;               z_ion      I     int     charge of recombined ion
;               te         I     real()  array of electron temperatures (K)
;               dens       I     real()  array of electron densities (cm^-3)
;               coef       O     real(,) array of radiative recom. coeffts (cm^3 s^-1)
;                                         1st dim: electron temperature
;                                         2nd dim: electron density
; OPTIONAL   :  config_z   I     str     full configuration string of final lowest
;                                          state (defaults to adf00 ground config.)
;               config_z1  I     str     full configuration string of initial state
;                                          (defaults to adf00 ground)
;               ionpot_z   I     real    ionisation potential of final lowest state
;                                        to ionised ion ground(Ryd)(defaults to adf00)
;               excpot_z1  I     real    excitation energy of initial state above
;                                          ionised ion ground (Ryd)(defaults to zero)
;               case_rr    I     str     selected approximation case (‘case a’,
;                                          ’case b’ ’case c’).(defaults to ’case b’).
;               crr        I     real    scaling factor for lowest complex capture
;                                          (defaults to 1.0)
;               derr       I     real    Temperature scaling factor for lowest complex
;                                          capture (defaults to 1.0)
;
; KEYWORDS   :  ev         I      -      switch ionpot_z, excpot_z1 and te[] to elec.
;                                          volts
;               help       I      -      prints help to screen
;
; NOTES      :
;
;
; AUTHOR     :  Hugh Summers
;
; DATE       :  28-11-2008
;
;
; MODIFIED:
;       1.1     Hugh Summers
;                 - first release
;
; VERSION:
;       1.1     28-11-2008
;
;-
;-----------------------------------------------------------------------

PRO alf_r_tot, z0_nuc     =  z0_nuc,     $
               z_ion   =  z_ion, $
               te   =  te, $
```

```
                dens   =  dens, $
                coef   =  coef, $
                config_z   =  config_z ,  $
                config_z1  =  config_z1,  $
                ionpot_z   =  ionpot_z,   $
                excpot_z1  =  excpot_z1,  $
                case_rr   =  case_rr, $
                crr   =  crr, $
                derr   =  derr, $
                ev   =  ev   $
                help   =  help  $


;-----------------------------------------------------------------------------
```

Notes:

alf_d_bgf.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  alf_d_bgf
;
; PURPOSE    :  calculates total zero-density dielectronic recombination
;               coefficients in the Burgess General formula approximation
;               to an ion at a set of electron temperatures. A low precision
;               finite density correction may be applied.
;
; ARGUMENTS  :  Arguments are non-positional named parameters.   All output
;               arguments will be defined appropriately.
;
;               NAME      I/O    TYPE    DETAILS
; REQUIRED   :  z0_nuc     I     int     nuclear charge of element
;               z_ion      I     int     charge of recombined ion
;               deij       I     real()  set of parent transition energies (Ryd)
;                                        (Note: absolute energy - not z-scaled)
;               fij        I     real()  set of parent upward oscillator strengths
;                                        from the parent ground state
;               te         I     real()  array of electron temperatures (K)
;               coef_tot   O     real()  array of dielectronic recom. coeffts (cm^3 s^-1)
;
; OPTIONAL   :  dens       I     real    electron density (cm^-3)
;                                        (faults if delta_nc not also set)
;               delta_nc   I     int     parent n-shell transition ( 0 or > 0)
;
; KEYWORDS   :  ev         I      -      switch deij[] and te[] to elec. volts
;               help       I      -      prints help to screen
;
; NOTES      :
;
;
```

```
; AUTHOR     :  Hugh Summers
;
; DATE       :  21-05-2009
;
;
; MODIFIED:
;       1.1     Hugh Summers
;                 - first release
;
; VERSION:
;       1.1     21-05-2009
;
;-
;-----------------------------------------------------------------------

PRO alf_d_bgf, z0_nuc     =  z0_nuc,     $
               z_ion   =  z_ion, $
               deij   =  deij, $
               fij   =  fij, $
               te   =  te, $
               dens   =  dens, $
               delta_nc   =  delta_nc, $
               coef_tot   =  coef_tot, $
               ev   =  ev,   $
               help   =  help


;-----------------------------------------------------------------------------
```

Notes:

alf_d_bgp.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  alf_d_bgp
;
; PURPOSE    :  calculates total zero density  and n-shell partial
;               dielectronic recombination coefficients in the Burgess
;               General Program approximation to an ion at a set of
;               electron temperatures.  A density dependent cut-off on the total
;               may be included.
;
; ARGUMENTS  :  Arguments are non-positional named parameters.   All output
;               arguments will be defined appropriately.
;
;               NAME      I/O   TYPE  DETAILS
; REQUIRED   :  z0_nuc     I    int  nuclear charge of element
;               z_ion      I    int  charge of recombined ion
;               ep         I    real  parent transition energy (cm-1)
;               fp         I    real  parent transition oscillator strength
```

```
;
;               te         I    real()   vector of electron temperatures (K)
;                                        1st dim: electron temperature index
; OPTIONAL      np         I    int      upper n-shell of parent transition
;               lp         I    int      upper l-shell of parent transition
;               ng         I    int      lower n-shell of parent transition
;                                        (belongs to the ground configuration)
;               lg         I    int      lower l-shell of parent transition
;                                        (belongs to the ground configuration)
;               cor        I    real()   Bethe partial wave correction factors
;                                        1st dim: partial wave index (=l+1)
;               df         I    real     Bethe global adjustment parameter
;                                        (defaults to 0.0)
;               nmin       I    int      lowest accessible n-shell by DR
;               def_nmin   I    real     quantum defect for lowest accesible n-shell by DR
;                                        (defaults to 0.0)
;               nrep       I    int()    representative n-shells
;                                        1st dim: representative n-shell index
;               phfrac     I    real     phase space factor for lowest accessible n-shell
;                                        (default to 1.0)
;               corfac     I    real     Bethe global adjustment parameter
;                                        (default to 0.0)
;               dens       I    real     electron density for n-shell cutoff in total
;                                        coefficient(cm-3)
;                                        (default to 0.0)
;
;               coef_tot   O    real()   vector of total DR coeffts (cm^3 s^-1) including
;                                        density cut-off of sum if dens set otherwise
;                                        zero density sum.
;                                        1st dim: electron temperature index
;               coef_n     O    real(,)  array of partial n-shell recom. coeffts (cm^3 s^-1)
;                                        to representative n-shells
;                                        1st dim: representative n-shell index
;                                        2nd dim: electron temperature index
;
; KEYWORDS   :  ev         I     - switch  te[] to elec. volts
;               help       I     - prints help to screen
;
; NOTES      :
;              1.  np,lp,ng,lg required if cor not set.  Must have lp=lg+/-1
;              2.  If cor not set, fortran default values are used based on np,lp,ng,lg
;              3.  df default to 0.0.  fortran default value used if cor not set
;              4.  nrep defaults to standard ADAS fortran internal value if not set.
;                  If not set, nmin is required.  If nrep and nmin supplied must have
;                  nmin=nrep(1)
;              5.  coef_n is not returned by fortran subroutine if nrep not set
;              6.  Associated fortran subroutine has temperatures in eV.
;
;
; AUTHOR     :  Hugh Summers
;
; DATE       :  29-05-2009
;
;
; MODIFIED:
;       1.1     Hugh Summers
;                 - first release
```

```
;
; VERSION:
;       1.1     29-05-2009
;
;-
;-----------------------------------------------------------------------

PRO alf_d_bgp, z0_nuc     =  z0_nuc,     $
               z_ion   =  z_ion, $
               ep   =  ep, $
               fp   =  fp, $
               te   =  te, $
               np   =  np, $
               lp   =  lp, $
               ng   =  ng, $
               lg   =  lg, $
               cor   =  cor, $
               df   =  df, $
               nmin   =  nmin, $
               def_nmin   =  def_nmin, $
               phfrac   =  phfrac, $
               corfac   =  corfac, $
               nrep   =  nrep, $
               te   =  te=, $
               dens   =  dens, $
               coef_tot   =  coef_tot, $
               coef_n   =  coef_n, $
               ev   =  ev,   $
               help   =  help


;-------------------------------------------------------------------------------
```

Notes:

alf_d_bbgp.pro

```
;-----------------------------------------------------------------------
;+
; PROJECT    :  ADAS
;
; NAME       :  alf_d_bbgp
;
; PURPOSE    :  calculates n-shell and nl-shell finite density
;               dielectronic recombination coefficients in the Burgess
;               Bethe General Program approximation to an ion at a set of
;               electron temperatures and electron densities.
;
; ARGUMENTS  :  Arguments are non-positional named parameters.   All output
;               arguments will be defined appropriately.
;
;               NAME      I/O   TYPE  DETAILS
; REQUIRED   :  z0_nuc     I    int  nuclear charge of element
```

```
;               z_ion      I    int  charge of recombined ion
;               adf46_file I    str      driver adf46 data set
;               te         I    real()   array of electron temperatures (K)
;               dens       I    real()   array of electron densities (cm-3)
;                                          (defaults to zero density case)
; OPTIONAL      zeff       I    real     effective charge of thermal ion colliders
;                                          (defaults to 1.0)
;               ams_zeff   I    real     effective mass of thermal ion colliders
;                                          (defaults to 2.0)
;               nrep       I    int()    representative n-shells
;                                          (defaults to standard ADAS set)
;               l_bound    I    int      highest l-shell for output coef_nl
;                                          (defaults to 10)
;               coef_n     O    real(,,) array of dielect. recom. coeffts (cm^3 s^-1)
;                                         1st dim: electron temperature
;                                         2nd dim: electron density
;                                         3rd dim: representative n-shell index
;               coef_nl    O    real(,,,)array of dielect. recom. coeffts (cm^3 s^-1)
;                                         1st dim: electron temperature
;                                         2nd dim: electron density
;                                         2nd dim: representative n-shell index
;                                         3rd dim: l-shell index
; KEYWORDS   :  ev         I     - switch te[] to elec. volts
;               help       I     - prints help to screen
;
; NOTES      :
;
;
; AUTHOR     :  Hugh Summers
;
; DATE       :  28-11-2008
;
;
; MODIFIED:
;       1.1     Hugh Summers
;                 - first release
;
; VERSION:
;       1.1     28-11-2008
;
;-
;-----------------------------------------------------------------------

PRO alf_d_bbgp, z0_nuc     =  z0_nuc,     $
                z_ion    =  z_ion,  $
                adf46_file =  adf46_file, $
                te    =  te,  $
                dens    =  dens,  $
                zeff    =  zeff,  $
                ams_zeff   =  ams_zeff,  $
                nrep    =  nrep,  $
                l_bound    =  l_bound,    $
                coef_n     =  coef_n,  $
                coef_nl    =  coef_nl,    $
                ev    =  ev,  $
                help    =  help
```

;---

Notes:

read_adf55.pro

;---

Notes:

run_adas407.pro

;---

Notes:

run_adas408.pro

;---

Notes:

run_adas316.pro

;---

Notes:

preview_natural_partition.pro

;---

Notes:

run_adas416.pro

```
;-------------------------------------------------------------------------
```

Appendix C

FORTRAN subroutines

Subroutine	Current location	Local checks			Central ADAS	
		Txt	Opr	Lnk	CVS	Rel
xxdata_00.for	/home/hps/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
xxdata_09.for	/home/hps/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
xxdata_11.for	/home/hps/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
xxdata_15.for	/home/hps/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
xxdata_23.for	/home/summers/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
xxdata_40.for	/home/hps/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
xxdata_46.for	/home/hps/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
xxdtes.for	/home/hps/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
xxcftr.for	/home/hps/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
g5dtes.for	/home/hps/adas_dev/fortran/adaslib/read_adf/	y	n	n	n	n
xxdrbf.for	/home/hps/adas_dev/fortran/adas7xx/adas708/	y	n	n	n	n
xxdrbp.for	/home/hps/adas_dev/fortran/adas7xx/adas708/	y	n	n	n	n
gxdrbp.for	/home/hps/adas_dev/fortran/adas7xx/adas708/	y	n	n	n	n
xxdraa.for	/home/hps/adas_dev/fortran/adas7xx/adas708/	y	n	n	n	n
g8bbgp.for	/home/hps/adas_dev/fortran/adas7xx/adas708/	y	n	n	n	n
xxwrto_09.for	/home/hps/adas_dev/fortran/adas7xx/adas708/	y	n	n	n	n

xxdata_00.for

```
       subroutine xxdata_00( iunit  , dsname ,
     &                       izdimd , iodimd , imdimd ,
     &                       esym   , iz0    , bwnoa  , eeva   ,
     &                       iorba  , na     , la     , iqa    ,
     &                       cstr_std        ,
     &                       imeta  , eevma           ,
     &                       iorbma , nma    , lma    , iqma   ,
     &                       cstrm_std       ,
     &                       lexist          , lresol
     &                     )
c
       implicit none
c-----------------------------------------------------------------------
c
c  ****************** fortran77 subroutine: xxdata_00 ******************
c
c  purpose:  to fetch data from an adf00 data set and detect its main
c            characteristics.
c
c                1. element symbol and nuclear charge
c                2. ionisation potentials (cm-1 and eV)
c                3. shell occupancies in the normal collating order
c
c  calling program: various
c
c           ionisation potential: eV
c           configuration:        standard form nlq (incl. integers
c                                 for n>9 and q>9 , lower case
c                                 letter for l and space separators)
c
c  subroutine:
c
c  input : (i*4)   iunit     = unit to which input file is allocated
c  input : (c*(*)) dsname    = name of opened data set on iunit
c  input : (i*4)   izdimd    = maximum nuclear charge
c  input : (i*4)   iodimd    = max. number of orbitals
c  input : (i*4)   imdimd    = max. number of metastables
c
c  output: (c*2)   esym      = element symbol.
c  output: (i*4)   iz0       =         nuclear charge read
c  output: (r*8)   bwnoa()   = ionisation potential (cm-1) of each stage
c                              1st dim: index = nuclear charge +1
c  output: (r*8)   eeva()    = ionisation potential (eV) of each stage
c                              1st dim: index = nuclear charge +1
c  output: (i*4)   iorba()   = number of orbital shells in configuration
c                              1st dim: index = nuclear charge +1
c  output: (i*4)   na(,)     = principal quantum number of shell
c                              1st dim: index = nuclear charge +1
c                              2nd dim: shell index
c  output: (i*4)   la(,)     = orbital ang. momentum qu. no. of shell
c                              1st dim: index = nuclear charge +1
c                              2nd dim: shell index
c  output: (i*4)   iqa(,)    = occupancy. of shell
```

```
c                               1st dim: index = nuclear charge +1
c                               2nd dim: shell index
c
c  output: (c*(*)) cstr_std()= configuration string in standard form
c                               1st dim: index = nuclear charge +1
c
c  output: (r*8)   eevma(,)  = excitation energy (eV) of each metastable
c                               1st dim: index = nuclear charge +1
c                               2nd dim: index = metastable index
c  output: (i*4)   iorbma(,) = number of orbital shells in metas. config.
c                               1st dim: index = nuclear charge +1
c                               2nd dim: index = metastable index
c  output: (i*4)   nma(,,)   = principal quantum number of metas.shell
c                               1st dim: index = nuclear charge +1
c                               2nd dim: shell index
c                               3rd dim: index = metastable index
c  output: (i*4)   lma(,,)   = orbital ang. mom. qu. no. of metas. shell
c                               1st dim: index = nuclear charge +1
c                               2nd dim: shell index
c                               3rd dim: index = metastable index
c  output: (i*4)   iqma(,,)  = occupancy. of metas. shell
c                               1st dim: index = nuclear charge +1
c                               2nd dim: shell index
c                               3rd dim: index = metastable index
c
c  output: (c*(*)) cstrm_std(,)=meta. config. string in standard form
c                               1st dim: index = nuclear charge +1
c                               2nd dim: index = metastable index
c
c  output: (l*4)   lexist  = .true.  => ionisation potential present
c                          = .false. => not present
c  output: (l*4)   lresol  = .true.  => metastable resolved adf00 file
c                          = .false. => not metastable resolved adf00
c
c
c routines:
c          routine    source    brief description
c          ------------------------------------------------------------
c          i4unit     adas      fetch unit number for output of messages
c          i4fctn     adas      converts from char. to integer  variable
c          xxslen     adas      finds string length excluding leading and
c                               trailing blanks
c          xxword     adas      parses a string into separate words
c                               for ' ()<>{}' delimiters
c          xxcase     adas      changes a string to upper or lower case
c          xfesym     adas      obtain element symbol from nuclear charge
c          xfelem     adas      obtain element name from nuclear charge
c          xxterm     adas      terminate program with a message
c
c
c author:  Hugh Summers, University of Strathclyde
c          JA7.08
c          tel. 0141-548-4196
c
c date:    27/04/04
c
c update:  15/12/06  H. P. Summers - extended to handle metastable resolved
```

```
c                                  adf00 files
c
c
c version: 1.1                          date: 27-04-04
c modified: H.P. Summers
c               - first version
c
c version: 1.2                          date: 05-01-07
c modified: H. P. Summers
c               - extended to handle metastable resolved
c                                  adf00 files
c
c
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      integer   nodim
c-----------------------------------------------------------------------
      real*8    dzero
c-----------------------------------------------------------------------
      parameter( nodim = 60 , dzero = 1.0d-30 )
c-----------------------------------------------------------------------
      integer*4  iunit    , izdimd   , iodimd   , imdimd
      integer*4  i4unit   , i4fctn   , iabt     , iz0
      integer*4  istart   , istop    , i        , j       , k        ,
     &           in       , n        , iq       , ind     , m
      integer*4  nfirst   , iwords   , maxwrd   , icurrent, nmet
c-----------------------------------------------------------------------
      integer*4  iorba(izdimd)     , na(izdimd,iodimd)     ,
     &           la(izdimd,iodimd) , iqa(izdimd,iodimd)
      integer*4  ifirst(nodim)     , ilast(nodim)
      integer*4  imeta(iodimd)
      integer*4  iorbma(izdimd,imdimd) ,
     &           nma(izdimd,iodimd,imdimd)    ,
     &           lma(izdimd,iodimd,imdimd)    ,
     &           iqma(izdimd,iodimd,imdimd)
c-----------------------------------------------------------------------
      real*8    bwnoa(izdimd)   , eeva(izdimd)  , eevma(izdimd,iodimd)
c-----------------------------------------------------------------------
      character dsname*(*), string*240
      character esym*2    , xfelem*12  , xfesym*2
      character c3*3      , c12*12
      character c80_1*80  , c80_2*80   , c80_3*80   , element*12
c-----------------------------------------------------------------------
      character cstr_std(izdimd)*(*)
      character cstrm_std(izdimd,imdimd)*(*)
c-----------------------------------------------------------------------
      character  c_nshl(35)*1   , c_lshl(22)*1  , c_qshl(36)*1
      character  c_scol(45)*2
c-----------------------------------------------------------------------
      logical   lexist          , lresol
c-----------------------------------------------------------------------
       data       c_scol /'1s','2s','2p','3s','3p','3d','4s','4p',
     &                    '4d','4f','5s','5p','5d','5f','5g','6s',
     &                    '6p','6d','6f','6g','6h','7s','7p','7d',
     &                    '7f','7g','7h','7i','8s','8p','8d','8f',
     &                    '8g','8h','8i','8j','9s','9p','9d','9f',
     &                    '9g','9h','9i','9j','9k'/
```

```
       data       c_nshl /'1','2','3','4','5','6','7','8','9',
     &                    'a','b','c','d','e','f','g','h','i',
     &                    'j','k','l','m','n','o','p','q','r',
     &                    's','t','u','v','w','x','y','z'/
       data       c_lshl /'s','p','d','f','g','h','i','j','k',
     &                    'l','m','n','o','q','r','t','u','v',
     &                    'w','x','y','z'/
       data       c_qshl /'0','1','2','3','4','5','6','7','8',
     &                    '9','a','b','c','d','e','f','g','h',
     &                    'i','j','k','l','m','n','o','p','q',
     &                    'r','s','t','u','v','w','x','y','z'/
c-----------------------------------------------------------------------
```

Notes:

xxdata_11.for

```
      subroutine  xxdata_11( iunit  , iclass ,
     &                       isdimd , iddimd , itdimd ,
     &                       ndptnl , ndptn  , ndptnc , ndcnct ,
     &                       iz0    , is1min , is1max ,
     &                       nptnl  , nptn   , nptnc  ,
     &                       iptnla , iptna  , iptnca ,
     &                       ncnct  , icnctv ,
     &                       iblmx  , ismax  , dnr_ele, dnr_ams,
     &                       isppr  , ispbr  , isstgr ,
     &                       idmax  , itmax  ,
     &                       ddens  , dtev   , drcof  ,
     &                       lres   , lstan  , lptn
     &                     )
       implicit none
c-----------------------------------------------------------------------
c
c  ****************** fortran77 subroutine: xxdata_11 ******************
c
c  purpose: to read a complete  adf11 file, check its class and
c           determine its standard, resolved and partition organisation.
c
c  calling program: various
c
c  notes:    (1) A 'standard' adf11 file contains gcr data between one
c                whole ionisation stage and another whole ionisation
c                stage.
c                A 'resolved' (or partial) adf11 file contains gcr data
c                between a set of metastables of one ionisation stage
c                and a set of metastables of another ionisation stage.
c                A resolved file is distinguished from a standard file
c                by the presence of a 'connection vector' in the adf11
c                data file header lines.
c                The connection vector specifies the number of meta-
c                stables in each ionisation stage which are coupled
c                together by gcr data.
c            (2) A 'partitioned' adf11 file contains gcr data between
```

```
c               clumps of ionisation stages or metastables or comb-
c               inations of the two called 'partitions'.
c               A 'partition level' is a specification of the
c               partitions which span all the ionisation stages (and
c               metastables) of an element.  Successive partition
c               levels give a heirarchy corresponding to larger
c               partitions and greater clumping.
c               A 'superstage' is a set of partitions which are close-
c               coupled.
c               There are thus equivalences :
c                       ionisation stage   -   superstage
c                       metastable         -   partition
c                       ion charge         -   superstage index
c               A partitioned adf11 file may be standard (with each
c               superstage comprising only one partition) or resolved.
c               A partitioned file is distinguished by the presence of
c               'partition specification block' in the adf11 data
c               file header lines.
c           (3) When  a partition specification block is present, it
c               should be ordered from the highest partition level
c               index to lowest partition level index.  Thus the first
c               partition in the partition block has the least number
c               of partitions and the last has the greatest number.
c           (4) Twelev classes of adf11 data file may be read by the
c               subroutine as follow:
c
c             class index type   GCR data content
c             ---------------  ----------------
c                 1   acd   recombination coeffts
c                 2   scd   ionisation coeffts
c                 3   ccd   CX recombination coeffts
c                 4   prb   recomb/brems power coeffts
c                 5   prc   CX power coeffts
c                 6   qcd   base meta. coupl. coeffts
c                 7   xcd   parent meta. coupl. coeffts
c                 8   plt   low level line power coeffts
c                 9   pls   represent. line power coefft
c                10   zcd   effective charge
c                11   ycd   effective squared charge
c                12   ecd   effective ionisation potential
c
c           (5) A resolved adf11 file, with a connection vector, has a set
c               of names and pointers at precise positions in the data file
c               which are recognised.
c               The names are different for partitioned and unpartitioned
c               data files as follow:
c
c                      file      unpartitioned         partitioned
c                      class        names                 names
c
c                      (all)         z1                   s1
c
c                        (indices 1 and 2)     (indices 1 and 2)
c                   ----   ----     ----         ----  ----
c                   acd        iprt     igrd         ispp   ispb
c                   scd        iprt     igrd         ispp   ispb
c                   ccd        iprt     igrd         ispp   ispb
```

```
c                    prb        iprt                  ispp
c                    prc        iprt                  ispp
c                    qcd        igrd     jgrd         ispb   jspb
c                    xcd        iprt     jprt         ispp   jspp
c                    plt        igrd           ispb
c                    pls        igrd           ispb
c                    zcd        igrd           ispb
c                    ycd        igrd           ispb
c                    ecd        igrd           ispb
c
c             (6) In partitioned nomenclature: s=superstage; p=partition;
c                 b=base (current superstage), p=parent (next up super-
c                 stage), c=child (next down superstage). Thus arrays
c                 'iprtr' and 'igrd' in old notation are now substituted
c                 by 'isppr' and 'ispbr' respectively internally and in
c                 external naming.
c
c
c  subroutine:
c
c  input : (i*4)  iunit     = unit to which input file is allocated
c  input : (i*4)  iclass    = class of data (1 - 12 ):
c                               1-acd, 2-scd, 3-ccd, 4-prb, 5-prc
c                               6-qcd, 7-xcd, 8-plt, 9-pls,10-zcd
c                              11-ycd,12-ecd
c
c  input : (i*4)  isdimd    = maximum number of (sstage, parent, base)
c                             blocks in isonuclear master files
c  input : (i*4)  iddimd    = maximum number of dens values in
c                             isonuclear master files
c  input : (i*4)  itdimd    = maximum number of temp values in
c                             isonuclear master files
c  input : (i*4)  ndptnl    = maximum level of partitions
c  input : (i*4)  ndptn     = maximum no. of partitions in one level
c  input : (i*4)  ndptnc    = maximum no. of components in a partition
c  input : (i*4)  ndcnct    = maximum number of elements in connection
c                             vector
c
c  output: (i*4)  iz0       = nuclear charge
c  output: (i*4)  is1min    = minimum ion charge + 1
c                             (generalised to connection vector index)
c  output: (i*4)  is1max    = maximum ion charge + 1
c                             (note excludes the bare nucleus)
c                             (generalised to connection vector index
c                              and excludes last one which always remains
c                              the bare nucleus)
c  output: (i*4)  nptnl     = number of partition levels in block
c  output: (i*4)  nptn()    = number of partitions in partition level
c                             1st dim: partition level
c  output: (i*4)  nptnc(,)  = number of components in partition
c                             1st dim: partition level
c                             2nd dim: member partition in partition level
c  output: (i*4)  iptnla()  = partition level label (0=resolved root,1=
c                                                    unresolved root)
c                             1st dim: partition level index
c  output: (i*4)  iptna(,)  = partition member label (labelling starts at 0)
c                             1st dim: partition level index
```

```
c                              2nd dim: member partition index in partition
c                              level
c  output: (i*4)  iptnca(,,)= component label (labelling starts at 0)
c                              1st dim: partition level index
c                              2nd dim: member partition index in partition
c                              level
c                              3rd dim: component index of member partition
c  output: (i*4)  ncnct      = number of elements in connection vector
c  output: (i*4)  icnctv()   = connection vector of number of partitions
c                              of each superstage in resolved case
c                              including the bare nucleus
c                              1st dim: connection vector index
c
c  output: (i*4)  iblmx      = number of (sstage, parent, base)
c                              blocks in isonuclear master file
c  output: (i*4)  ismax      = number of charge states
c                              in isonuclear master file
c                              (generalises to number of elements in
c                               connection vector)
c  output: (c*12) dnr_ele    = CX donor element name for iclass = 3 or 5
c                              (blank if unset)
c  output: (r*8)  dnr_ams    = CX donor element mass for iclass = 3 or 5
c                              (0.0d0 if unset)
c  output: (i*4)  isppr()    = 1st (parent) index for each partition block
c                              1st dim: index of (sstage, parent, base)
c                                       block in isonuclear master file
c  output: (i*4)  ispbr()    = 2nd (base) index for each partition block
c                              1st dim: index of (sstage, parent, base)
c                                       block in isonuclear master file
c  output: (i*4)  isstgr()   = s1 for each resolved data block
c                              (generalises to connection vector index)
c                              1st dim: index of (sstage, parent, base)
c                                       block in isonuclear master file
c
c  output: (i*4)  idmax      = number of dens values in
c                              isonuclear master files
c  output: (i*4)  itmax      = number of temp values in
c                              isonuclear master files
c  output: (r*8)  ddens()    = log10(electron density(cm-3)) from adf11
c  output: (r*8)  dtev()     = log10(electron temperature (eV) from adf11
c  output: (r*8)  drcof(,,) = if(iclass <=9):
c                                 log10(coll.-rad. coefft.) from
c                                 isonuclear master file
c                              if(iclass >=10):
c                                 coll.-rad. coefft. from
c                                 isonuclear master file
c                              1st dim: index of (sstage, parent, base)
c                                       block in isonuclear master file
c                              2nd dim: electron temperature index
c                              3rd dim: electron density index
c
c  output: (l*4)  lres       = .true.  => partial file
c                            = .false. => not partial file
c  output: (l*4)  lstan      = .true.  => standard file
c                            = .false. => not standard file
c  output: (l*4)  lptn       = .true.  => partition block present
c                            = .false. => partition block not present
```

```
c
c  routines:
c          routine    source    brief description
c          ------------------------------------------------------------
c          i4unit     adas      fetch unit number for output of messages
c          i4fctn     adas      convert string to integer form
c          xfelem     adas      return element name given nuclear charge
c          xxword     adas      extract position of number in buffer
c          xxslen     adas      find string less front and tail blanks
c          xxcase     adas      convert a string to upper or lower case
c          xxrptn     adas      analyse an adf11 file partition block
c
c author:  h. p. summers, university of strathclyde
c          ja7.08
c          tel. 0141-548-4196
c
c date:    04/10/06
c
c version: 1.1 date: 04/10/2006
c modified: hugh summers
c - first edition.
c
c version: 1.2 date: 21/01/2007
c modified: Allan Whiteford
c - Commented out warning about lack of iclass,
c                 all of the present ADAS files do not contain
c                 this information
c                 (first commit to CVS)
c
c version: 1.3 date: 08/03/2007
c modified: Hugh Summers
c - adjustments for revised ecd formats.
c                 charge exchange donor/donor mass checks and
c                 dnr_ele, dnr_ams added to parameter return.
c
c-----------------------------------------------------------------------
      integer   nddash    , idword      , ndstack  , ndonors
c-----------------------------------------------------------------------
      character csrch1*4   , csrch2*4     , csrch3*4
      character csrch4*4   , csrch5*4
      character cdash*8    , cpart*3
c-----------------------------------------------------------------------
      parameter ( nddash = 6  , idword = 256 , ndstack = 40 )
      parameter ( csrch1 = 'iprt' , csrch2 = 'igrd'  ,csrch3 = '---/')
      parameter ( csrch4 = 'ispp' , csrch5 = 'ispb')
      parameter ( cdash = '--------' , cpart = '//#' )
      parameter ( ndonors = 4 )
c-----------------------------------------------------------------------
      integer   iunit     , i4unit   , i4fctn
      integer   iz0       , is1min   , is1max
      integer   iddimd    , itdimd   , isdimd
      integer   ndptnl    , ndptn    , ndptnc   , ndcnct
      integer   iblmx     , ismax    , itmax    , idmax
      integer   ischk
      integer   iclass    , iclass_file
      integer   i         , ic       , it         , id
      integer   icptn     , iabt     , iabt1      , iabt2    ,
```

```
     &          j         , ndash_line
      integer   nptnl     , ncnct    , ncptn_stack
      integer   nfirst    , iwords   , nwords   , ifirst     , ilast
c-----------------------------------------------------------------------
      real*8    dnr_ams   , dmass
c-----------------------------------------------------------------------
      character cstrg*80  , cstrgl*80, cterm*80   ,  chindi*4
      character xfelem*12 , str12*12 , dnr_ele*12
c-----------------------------------------------------------------------
      logical   lres      , lstan     , lptn      , lresol
      logical   lptn_old  , lwarn     , ldonor    , ldmass
c-----------------------------------------------------------------------
      integer   idash_linea(nddash)
      integer   nptn(ndptnl)          , nptnc(ndptnl,ndptn)
      integer   iptnla(ndptnl)        , iptna(ndptnl,ndptn)
      integer   iptnca(ndptnl,ndptn,ndptnc)
      integer   icnctv(ndcnct)
      integer   isstgr(isdimd)
      integer   ifirsta(idword)       , ilasta(idword)
      integer   ispbr(isdimd)         , isppr(isdimd)
c-----------------------------------------------------------------------
      real*8    ddens(iddimd)         , dtev(itdimd)
      real*8    drcof(isdimd,itdimd,iddimd)
c-----------------------------------------------------------------------
      character cclass(12)*4          , cpatrn(12)*4
      character cptrn1(12)*4          , cptrn2(12)*4
      character cptn_stack(ndstack)*80
      character cdonors(ndonors)*12
c-----------------------------------------------------------------------
      data      cterm /'-----------------------------------------------
     &--------------------------------'/
      data      cclass/'/ACD','/SCD','/CCD','/PRB','/PRC',
     &               '/QCD','/XCD','/PLT','/PLS','/ZCD',
     &               '/YCD','/ECD'/
      data      cptrn1/'iprt','iprt','iprt','iprt','iprt',
     &               'igrd','iprt','igrd','igrd','igrd',
     &               'igrd','igrd'/
      data      cptrn2/'ispp','ispp','ispp','ispp','ispp',
     &               'ispb','ispp','ispb','ispb','ispb',
     &               'ispb','ispb'/
      data      cdonors/'HYDROGEN    ','DEUTERIUM   ','TRITIUM     ',
     &                'HELIUM      '/
c-----------------------------------------------------------------------
```

Notes:

xxdata_15.for

```
       subroutine xxdata_15( iunit  , dsname ,
     &                       nstore , ntdim  , nddim  ,
     &                       ndptnl , ndptn  , ndptnc , ndcnct ,
     &                       ndstack, ndcmt ,
     &                       iz0    , is     , is1    , esym   ,
     &                       nptnl  , nptn   , nptnc  ,
     &                       iptnla , iptna  , iptnca ,
     &                       ncnct  , icnctv ,
     &                       ncptn_stack     , cptn_stack      ,
     &                       lres   , lptn   , lcmt   , lsup   ,
     &                       nbsel  , isela  ,
     &                       cwavel , cfile  , ctype  , cindm  ,
     &                       wavel  , ispbr  , isppr  , isstgr , iszr  ,
     &                       ita    , ida    ,
     &                       teta   , teda   ,
     &                       pec    , pec_max,
     &                       ncmt_stack      , cmt_stack
     &                     )
       implicit none
c-----------------------------------------------------------------------
c
c  ***************** fortran77 subroutine: xxdata_15 *******************
c
c  purpose:  To fetch  data  from  an input photon emissivity file
c            for a given emitting element superstage .
c
c  calling programs: adas416/dxdata_15
c
c  data:    Up to 'nstore' sets (data-blocks) of data may be  read from
c           the file - each block forming a complete set of photon
c           emissivity coefft. values for given temp/density grid.
c           Each data-block  is  analysed independently of any  other
c           datablock.
c
c           the units used in the data file are taken as follows:
c
c           temperatures : ev
c           densities    : cm-3
c           pec          : phot. cm3 s-1
c
c  subroutine:
c
c  input : (i*4)  iunit    = unit to which input file is allocated.
c          (i*4)  dsname   = name of opened data set on iunit
c
c          (i*4)  nstore   = maximum number  of  input data-blocks  that
c                            can be stored.
c          (i*4)  ntdim    = max number of electron temperatures allowed
c          (i*4)  nddim    = max number of electron densities    allowed
c          (i*4)  ndptnl   = maximum level of partitions
c          (i*4)  ndptn    = maximum no. of partitions in one level
c          (i*4)  ndptnc   = maximum no. of components in a partition
c          (i*4)  ndcnct   = maximum number of elements in connection
c          (i*4)  ndstack  = maximum number of partition text lines
```

```
c          (i*4)  ndcmt    = maximum number of comment text lines
c                             vector
c output:  (i*4)  iz0      = read - emitting ion - nuclear charge
c          (i*4)  is       = read - emitting ion - charge
c                            (generalised to superstage label)
c          (i*4)  is1      = read - emitting ion - charge + 1
c                            (generalised to superstage index= is + 1)
c          (c*2)  esym     = read - emitting ion - element symbol
c
c          (i*4)  nptnl    = number of partition levels in block
c          (i*4)  nptn()   = number of partitions in partition level
c                            1st dim: partition level
c          (i*4)  nptnc(,) = number of components in partition
c                            1st dim: partition level
c                            2nd dim: member partition in partition level
c          (i*4)  iptnla() = partition level label (0=resolved root,1=
c                                                   unresolved root)
c                            1st dim: partition level index
c          (i*4)  iptna(,) = partition member label (labelling starts at 0)
c                            1st dim: partition level index
c                            2nd dim: member partition index in partition
c                            level
c          (i*4)  iptnca(,,)= component label (labelling starts at 0)
c                            1st dim: partition level index
c                            2nd dim: member partition index in partition
c                            level
c                            3rd dim: component index of member partition
c          (i*4)  ncnct     = number of elements in connection vector
c          (i*4)  icnctv()  = connection vector of number of partitions
c                             of each superstage in resolved case
c                             including the bare nucleus
c                             1st dim: connection vector index
c          (i*4)  ncptn_stack = number of text lines in partition block
c          (c*80) cptn_stack()= text lines in partition block
c                               1st dim: text line index (1->ncptn_stack)
c
c          (l*4)  lres      = .true.  => partial file
c                           = .false. => not partial file
c          (l*4)  lptn      = .true.  => partition block present
c                           = .false. => partition block not present
c          (l*4)  lcmt      = .true.  => comment text block present
c                           = .false. => comment text block not present
c          (l*4)  lsup      = .true.  => ss use of filmem field
c                           = .false. => old use of filmem field
c
c          (i*4)  nbsel     = number of data-blocks accepted & read in.
c          (i*4)  isela()   = read - data-set data-block entry indices
c                            dimension: data-block index
c
c          (c*10) cwavel() = wavelength string (angstroms)
c                            1st dim: data-block index
c          (c*8)  cfile()  = specific ion file source string in older
c                            forms.  Field not present in superstage
c                            version, but reused for added information
c                            1st dim: data-block index
c          (c*8)  ctype()  = data type string
c                            1st dim: data-block index
```

```
c          (c*2)  cindm()  = metastable index string
c                            1st dim: data-block index
c
c          (r*8)  wavel()   = wavelength (angstroms)
c                            dimension: data-block index
c          (i*4)  isppr()   = parent index for each line block
c                            1st dim: index of block in adf15 file
c          (i*4)  ispbr()   = base index for each line block
c                            1st dim: index of block in adf15 file
c          (i*4)  isstgr()  = s1 for each resolved data block
c                            1st dim: index of block in adf15 file
c          (i*4)  iszr()    = ion charge relating to each line
c                            1st dim: index of block in adf15 file
c
c          (i*4)  ita()     = number of electron temperatures
c                            dimension: data-block index
c          (i*4)  ida()     = read - number of electron densities
c                            1st dim: data-block index
c
c          (r*8)  teta(,)   =  electron temperatures (units: ev)
c                            1st dim: electron temperature index
c                            2nd dim: data-block index
c          (r*8)  teda(,)   = electron densities (units: cm-3)
c                            1st dim: electron density index
c                            2nd dim: data-block index
c
c          (r*8)  pec(,,)   = photon emissivity coeffts
c                            1st dim: electron temperature index
c                            2nd dim: electron density index
c                            3rd dim: data-block index
c          (r*8)  pec_max() = photon emissivity coefft. maximum
c                            as a function of Te at first Ne value
c                            1st dim: data-block index
c          (i*4)  ncmt_stack = number of text lines in comment block
c          (c*80) cmt_stack()= text lines in comment block
c                               1st dim: text line index (1->ncmt_stack)
c
c routine: (i*4)  i4eiz0   = function - (see routines section below)
c          (i*4)  i4fctn   = function - (see routines section below)
c          (i*4)  i4unit   = function - (see routines section below)
c          (i*4)  iblk     = array index: data-block index
c          (i*4)  itt      = array index: electron temperature index
c          (i*4)  itd      = array index: electron density     index
c          (i*4)  ntnum    = number of electron temperatures for current
c                            data-block
c          (i*4)  ndnum    = number of electron densities    for current
c                            data-block
c          (i*4)  iabt     = return code from 'i4fctn'
c          (i*4)  ipos1    = general use string index variable
c          (i*4)  ipos2    = general use string index variable
c
c          (l*4)  lbend    = identifies whether the last of the  input
c                            data sub-blocks has been located.
c                            (.true. => end of sub-blocks reached)
c
c          (c*1)  cslash   = '/' - delimiter for 'xxhkey'
c          (c*2)  c2       = general use two byte character string
```

```
c          (c*5)  ionnam   = emitting ion read from dataset
c          (c*6)  ckey1    = 'filmem' - input block header key
c          (c*4)  ckey2    = 'type  ' - input block header key
c          (c*4)  ckey3    = 'indm  ' - input block header key
c          (c*4)  ckey4    = 'isel  ' - input block header key
c          (c*80) c80      = general use 80 byte  character  string  for
c                            the input of data-set records.
c
c routines:
c          routine    source    brief description
c          ------------------------------------------------------------
c          i4eiz0     adas      returns z0 for given element symbol
c          i4fctn     adas      convert character string to integer
c          i4unit     adas      fetch unit number for output of messages
c          r8fctn     adas      convert string to real number
c          xxmkrp     adas      make up root partition text lines
c          xxcase     adas      convert a string to upper or lower case
c          xxhkey     adas      obtain key/response strings from text
c          xxrptn     adas      analyse an adf11 file partition block
c          xxword     adas      extract position of number in buffer
c          xxslen     adas      find string less front and tail blanks
c
c author:  h. p. summers
c          k1/1/57
c          jet ext. 4941
c
c date:    11/10/91
c
c update:  05/12/91 - pe briden: ionnam now allowed to occupy either
c                                4 or 5 spaces in the header.
c
c update:  23/04/93 - pe briden - adas91: added i4unit function to write
c                                         statements for screen messages
c
c update:  24/05/93 - pe briden - adas91: changed i4unit(0)-> i4unit(-1)
c
c update:  27/2/95  - l. jalota - idl_adas : increased size dsname for
c                                            use under unix systems
c
c unix-idl port:
c
c version: 1.2                          date: 23-1-96
c modified: tim hammond (tessella support services plc)
c               - corrected format statements for dsname length
c
c-----------------------------------------------------------------------
c
c
c notes: copied from e3data.for. this is v1.1 of xxdata_15.
c
c
c version  : 1.1
c date     : 12-04-2005
c modified : martin o'mullane
c              - first version
c
c version  : 1.2
```

```
c date     : 25-04-2005
c modified : martin o'mullane
c              - increase c3 to character*3 to permit more than
c                100 entries in adf15 file.
c
c version  : 1.3
c date     : 15-05-2006
c modified : Hugh Summers
c              - extended to operation with superstages and partitions.
c
c version : 1.4                          date: 03/01/2007
c modified: Hugh Summers
c               - remove redundant variables.
c
c-----------------------------------------------------------------------
      integer    idword    , idcnct  , iz0_max_res
c-----------------------------------------------------------------------
      parameter ( idword = 256 , idcnct = 100 )
      parameter ( iz0_max_res = 10 )
c-----------------------------------------------------------------------
      integer    i4eiz0    , i4fctn   , i4unit
      integer    iunit     , nstore   ,
     &           ntdim     , nddim    ,
     &           iz0       , is       ,
     &           is1       , nbsel
      integer    iblk      ,
     &           itt       , itd      ,
     &           ntnum     , ndnum    ,
     &           ipos1     , ipos2    , iabt
      integer    ndptnl    , ndptn    , ndptnc        , ndcnct
      integer    ndstack   , ndcmt
      integer    nptnl     , ncnct    , ncptn_stack   , ncmt_stack  ,
     &           iptnl
      integer    nfirst    , iwords   , nwords
      integer    i         , j        , ifirst        , ilast
      integer    max_indm  , indm
c-----------------------------------------------------------------------
      real*8     pmax      , r8fctn
c-----------------------------------------------------------------------
      logical    lbend
      logical    lptn      , lresol   , lres    , lsup
      logical    lcmt      , lptn_temp
c-----------------------------------------------------------------------
      character  dsname*80             , esym*2
      character  cslash*1              , colon*1
      character  c2*2                  , c3*3                ,
     &           ckey1*6               , ckey2*4             ,
     &           ckey3*4               , ckey4*4             ,
     &           ckey5*2               , ckey6*2             ,
     &           ckey7*2               , ckey8*2             ,
     &           ckey9*2               , ckey10*4            ,
     &           ckey11*4              ,
     &           ionnam*5              , c80*80              , cstrg*80
      character  cblnk8*8
c-----------------------------------------------------------------------
      integer    isela(nstore)         ,
     &           ita(nstore)           , ida(nstore)
```

```
      integer    nptn(ndptnl)          , nptnc(ndptnl,ndptn)
      integer    iptnla(ndptnl)        , iptna(ndptnl,ndptn)
      integer    iptnca(ndptnl,ndptn,ndptnc)
      integer    icnctv(ndcnct)
      integer    ifirsta(idword)       , ilasta(idword)
      integer    isstgr(nstore)        , iszr(nstore)
      integer    ispbr(nstore)         , isppr(nstore)
      integer    ncncta(iz0_max_res)   , icnctva(iz0_max_res,idcnct)
c-----------------------------------------------------------------------
      character  cindm(nstore)*2       , cfile(nstore)*8      ,
     &           ctype(nstore)*8       , cwavel(nstore)*10
      character  cptn_stack(ndstack)*80,cmt_stack(ndcmt)*80
c-----------------------------------------------------------------------
      real*8     teta(ntdim,nstore)    , teda(nddim,nstore)
      real*8     wavel(nstore)         , pec(ntdim,nddim,nstore)
      real*8     pec_max(nstore)
c-----------------------------------------------------------------------
      save       cslash                ,
     &           ckey1                 , ckey2                ,
     &           ckey3                 , ckey4
c-----------------------------------------------------------------------
      data       cslash / '/' /        , colon  / ':' /
      data       cblnk8 /'        '/
      data       ckey1  / 'filmem' /   , ckey2  / 'type'   /  ,
     &           ckey3  / 'indm'   /   , ckey4  / 'isel'   /  ,
     &           ckey5  /  'pl'    /   , ckey6  /  'ss'    /  ,
     &           ckey7  /  'pb'   /    , ckey8  /  'pp'    /  ,
     &           ckey9  /  'sz'   /    , ckey10 / 'ispb'   /  ,
     &           ckey11 / 'ispp'   /

      data  ncncta(1),(icnctva(1,i),i=1,2)   / 2,1,1/
      data  ncncta(2),(icnctva(2,i),i=1,3)   / 3,2,1,1/
      data  ncncta(3),(icnctva(3,i),i=1,4)   / 4,1,2,1,1/
      data  ncncta(4),(icnctva(4,i),i=1,5)   / 5,2,1,2,1,1/
      data  ncncta(5),(icnctva(5,i),i=1,6)   / 6,2,2,1,2,1,1/
      data  ncncta(6),(icnctva(6,i),i=1,7)   / 7,4,2,2,1,2,1,1/
      data  ncncta(7),(icnctva(7,i),i=1,8)   / 8,3,4,2,2,1,2,1,1/
      data  ncncta(8),(icnctva(8,i),i=1,9)   / 9,4,3,4,2,2,1,2,1,1/
      data  ncncta(9),(icnctva(9,i),i=1,10)  /10,2,4,3,4,2,2,1,2,1,1/
      data  ncncta(10),(icnctva(10,i),i=1,11)/11,2,2,4,3,4,2,2,1,2,1,1/
c-----------------------------------------------------------------------
```

Notes:

xxdata_23.for

```
      subroutine xxdata_23(iunit    ,
     &                     ndlev    , ndmet     , ndtem     , ndtext   ,
     &                     seq      , iz0       , iz        , iz1      ,
     &                     ctype    ,
     &                     bwno_f   , nlvl_f    , lmet_f    , lcstrg_f ,
     &                     ia_f     , code_f    , cstrga_f  ,
     &                     isa_f    , ila_f     , xja_f     , wa_f     ,
     &                     nmet_f   , imeta_f   ,
     &                     bwno_i   , nlvl_i    , lmet_i    , lcstrg_i ,
     &                     ia_i     , code_i    , cstrga_i  ,
     &                     isa_i    , ila_i     , xja_i     , wa_i     ,
     &                     nmet_i   , imeta_i   ,
     &                     nte_ion  , tea_ion   , lqred_ion , qred_ion ,
     &                     nf_a     , indf_a    , lyld_a    , yld_a    ,
     &                     nte_exc  , tea_exc   , lqred_exc , qred_exc ,
     &                     l_ion    , l_aug     , l_exc     ,
     &                     ntext    , ctext
     &                    )

      implicit none
c-----------------------------------------------------------------------
c  ******************* fortran77 subroutine: xxdata_23 *******************
c
c  purpose:  to fetch data from an adf23 data set.
c
c  input : (i*4)  iunit       = unit to which input file is allocated
c          (i*4)  ndlev       = maximum number of energy levels in
c                               either ion stage
c          (i*4)  ndmet       = maximum number of metastables
c          (i*4)  ndtem       = maximum number of temperatures
c          (i*4)  ndtext      = maximum number of comment text lines
c
c  output: (c*2)  seq         = iso-electronic sequence symbol
c          (i*4)  iz0         = nuclear charge
c          (i*4)  iz          = ionising ion charge
c          (i*4)  iz1         = ioniswd ion charge (=iz+1)
c          (c*2)  ctype       = adf23 file resol. ('ca', 'ls' or 'ic')
c          (r*8)  bwno_f        = ionis. poten. of ionised ion (cm-1)
c          (i*4)  nlvl_f        = number of levels of ionised ion (cm-1)
c          (l*4)  lmet_f      = .true.  => ionised metastables marked
c                               .false. => ionised metastables unmarked
c                                          (default action - mark ground)
c          (l*4)  lcstrg_f    = .true.  => standard config strings for
c                                          ionised ion states
c                               .false. => unreadable config string for
c                                          at least one ionised ion state
c          (i*4)  ia_f()      = index of ionised ion levels
c                               1st dim: ionised ion level index
c          (c*1)  code_f()    = met. or excit. DR parent marker (* or #)
c                               1st dim: ionised ion level index
c          (i*(*))cstrga_f()  = ionised ion configuration strings
c                               1st dim: ionised ion level index
c          (i*4)  isa_f()     = ionised ion level multiplicity
c                               1st dim: ionised ion level index
```

```
c          (i*4)  ila_f()     = ionised ion total orb. ang. mom.
c                                1st dim: ionised ion level index
c          (r*8)  xja_f()     = ionised ion level (stat wt-1)/2
c                                1st dim: ionised ion level index
c          (r*8)  wa_f()      = ionised ion level wave number (cm-1)
c                                1st dim: ionised ion level index
c          (i*4)  nmet_f      = number of ionised ion metastables
c          (i*4)  imeta_f()   = pointers to ionised metastables in full
c                                ionised ion state list
c                                1st dim: ionised metastable index
c          (r*8)  bwno_i        = ionis. poten. of ionising ion (cm-1)
c          (i*4)  nlvl_i        = number of levels of ionising ion (cm-1)
c          (l*4)  lmet_i      = .true.  => ionising metastable marked
c                               .false. => ionising metastables unmarked
c                                          (default action - mark ground)
c          (l*4)  lcstrg_i    = .true.  => standard config strings for
c                                          ionising ion states
c                               .false. => unreadable config string for
c                                          at least one ionising ion state
c          (i*4)  ia_i()      = index of ionising ion levels
c                                1st dim: ionising ion level index
c          (c*1)  code_i()    = met. or excit. DR parent marker (* or #)
c                                1st dim: ionising ion level index
c          (i*(*))cstrga_i()  = ionising ion configuration strings
c                                1st dim: ionising ion level index
c          (i*4)  isa_i()     = ionising ion level multiplicity
c                                1st dim: ionising ion level index
c          (i*4)  ila_i()     = ionising ion total orb. ang. mom.
c                                1st dim: ionising ion level index
c          (r*8)  xja_i()     = ionising ion level (stat wt-1)/2
c                                1st dim: ionising ion level index
c          (r*8)  wa_i()      = ionising ion level wave number (cm-1)
c                                1st dim: ionising ion level index
c          (i*4)  nmet_i      = number of ionising ion metastables
c          (i*4)  imeta_i()   = pointers to ionising metastables in full
c                                ionising ion state list
c                                1st dim: ionising metastable index
c          (i*4)  nte_ion()   = number of temperatures for direct ionis-
c                                ation data for initial metastable block
c                                1st dim: ionising ion metastable index
c          (r*8)  tea_ion(,)  = temperatures (K) for direct ionis-
c                                ation data for initial metastable block
c                                1st dim: ionising ion metastable index
c                                2nd dim: temperature index
c          (l*4)  lqred_ion(,)= .true. => direct ionisation data line
c                                         present for ionised ion state
c                               .false.=> data line not present for
c                                         ionised ion state.
c                                1st dim: ionising ion metastable index
c                                2nd dim: ionised ion state index
c          (r*8)  qred_ion(,,)= reduced direct ionisation rate coeffts.
c                                1st dim: ionising ion metastable index
c                                2nd dim: ionised ion state index
c                                3rd dim: temperature index
c          (i*4)  nf_a()      = number of Auger ionised ion final states
c                                1st dim: ionising ion metastable index
c          (i*4)  indf_a(,)   = Auger ionised ion final state
```

```
c                               1st dim: ionising ion metastable index
c                               2nd dim: final state index
c          (l*4)  lyld_a(,)   = .true. => Auger data for ionising ion excited state
c                               .false.=> no Auger data
c                               1st dim: ionising ion metastable index
c                               2nd dim: initial state index
c          (r*8)  yld_a(,,)   = Auger yields
c                               1st dim: ionising ion metastable index
c                               2nd dim: ionising ion excited state index
c                               3rd dim: ionised ion state index
c          (i*4)  nte_exc()   = number of temperatures for excitation
c                               data for initial metastable block
c                               1st dim: ionising ion metastable index
c          (r*8)  tea_exc(,)  = temperatures (K) for direct excit-
c                               ation data for initial metastable block
c                               1st dim: ionising ion metastable index
c                               2nd dim: temperature index
c          (l*4)  lqred_exc(,)= .true. => direct excitation data line
c                                         present for excited ion state
c                               .false.=> data line not present for
c                                         excited ion state.
c                               1st dim: ionising ion metastable index
c                               2nd dim: excited ionising ion state index
c          (r*8)  qred_exc(,,)= reduced excitation rate coeffts.
c                               1st dim: ionising ion metastable index
c                               2nd dim: excited ionising ion state index
c                               3rd dim: temperature index
c          (l*4)  l_ion()     = .true. => ionisation data present for metastable
c                               .false.=> ionisation data not present
c                               1st dim: ionising ion metastable index
c          (l*4)  l_aug()     = .true. => Auger data present for metastable
c                               .false.=> Auger data not present
c                               1st dim: ionising ion metastable index
c          (l*4)  l_exc()     = .true. => excitation data present for metastable
c                               .false.=> excitation data not present
c                               1st dim: ionising ion metastable index
c          (i*4)  ntext       = number of commment text lines
c          (c*80) ctext()     = comment text lines
c                               1st dim: index of text lines
c
c
c  routines:
c          routine    source    brief description
c          ------------------------------------------------------------
c          i4unit     adas      fetch unit number for output of messages
c          i4eiz0     adas      fetch nuclear charge for element symbol
c          xfesym     adas      fetch element symbol for nuclear charge
c          xxcase     adas      convert string to lower or upper case
c          xxhkey     adas      extract a key name value from a string
c          xxlast     adas      find last occurence of char in string
c          xxslen     adas      find first and last characters of string
c          xxdtes     adas      detect if config string is eissner/standard
c          xxcftr     adas      covert config string between eissner/standard
c
c  author:  hugh summers
c  date  :  30-05-2008
c
```

```
c
c  version  : 1.1
c  date     : 30-05-2008
c  modified : hugh summers
c               - first version
c
c
c-----------------------------------------------------------------------
      integer   nsym    , idlev
c-----------------------------------------------------------------------
      real*8    const1
c-----------------------------------------------------------------------
      parameter( nsym = 92 , idlev= 100 , const1=8065.541d0 )
c-----------------------------------------------------------------------
      integer   i4unit     , i4eiz0     , lenstr
      integer   iunit
      integer   ndlev      , ndmet      , ndtem    , ndtext
      integer   iz0        , iz         , iz1      , izf
      integer   nlvl_f     , nlvl_i
      integer   ind1       , ind2       , istart   , istop   ,
     &          i          , j          , indx_fb  , indx_lb
      integer   iprf       , it         , nte
      integer   imet_i     , imet_count , indi     , ilvl
      integer   iword      , nwords     ,
     &          ilen_index , ilen_cnfg  , ilen_s     ,
     &          ilen_l     , ilen_j     , ilen_wnf
      integer   nvlce      , ilen
      integer   nmet_i     , nmet_f
      integer   ntext
c-----------------------------------------------------------------------
      real*8    bwno_f       , bwno_i
c-----------------------------------------------------------------------
      character seq*2      , esym*2     , ctype*2    , xfesym*2
      character c10*10     , c22*22     , c80*80     , c256*256
      character cline*80   , clong*256
      character cbreak*1   , csrch*1
      character f_1004*41  , f_1005*29
c-----------------------------------------------------------------------
      logical   lmet_f     , lmet_i     , lcstrg_f   , lcstrg_i
      logical   lstan      , leiss      , lcheck
c-----------------------------------------------------------------------
      integer   ia_f(ndlev)
      integer   isa_f(ndlev)    , ila_f(ndlev)
      integer   ia_i(ndlev)
      integer   isa_i(ndlev)    , ila_i(ndlev)
      integer   ifirst(1)       , ilast(1)
      integer   nte_ion(ndmet)  , nte_exc(ndmet)     , nf_a(ndmet)
      integer   indf_a(ndmet,ndlev)
      integer   imeta_i(ndmet)  , imeta_f(ndmet)
c-----------------------------------------------------------------------
      real*8    xja_f(ndlev)    , wa_f(ndlev)
      real*8    xja_i(ndlev)    , wa_i(ndlev)
      real*8    tea_ion(ndmet,ndtem)  , qred_ion(ndmet,ndlev,ndtem)
      real*8    tea_exc(ndmet,ndtem)  , qred_exc(ndmet,ndlev,ndtem)
      real*8    yld_a(ndmet,ndlev,ndlev)
      real*8    duma(idlev)
c-----------------------------------------------------------------------
```

```
      character code_f(ndlev)*1 , cstrga_f(ndlev)*(*)
      character code_i(ndlev)*1 , cstrga_i(ndlev)*(*)
      character ckey(3)*6       , cans(3)*4
      character ctext(ndtext)*80
c-----------------------------------------------------------------------
      logical   l_ion(ndmet)     , l_aug(ndmet)      , l_exc(ndmet)
      logical   lqred_ion(ndmet,ndlev)
      logical   lqred_exc(ndmet,ndlev)
      logical   lyld_a(ndmet,ndlev)
c-----------------------------------------------------------------------
      data      ckey/'seq','nucchg','type'/
      data      cbreak,csrch/'/',')'/
c-----------------------------------------------------------------------
```

Notes:

xxdata_40.for

```
      subroutine xxdata_40( iunit  , dsname  ,
     &                      nstore , ndpix   , ntdim  , nddim  ,
     &                      ndptnl , ndptn   , ndptnc , ndcnct ,
     &                      ndstack, ndcmt   ,
     &                      iz0    , is      , is1    , esym   ,
     &                      nptnl  , nptn    , nptnc  ,
     &                      iptnla , iptna   , iptnca ,
     &                      ncnct  , icnctv  ,
     &                      ncptn_stack      , cptn_stack      ,
     &                      lres   , lptn    , lcmt   , lsup   ,
     &                      nbsel  , isela   ,
     &                      npixa  , cfile   , ctype  , cindm  ,
     &                      ispbr  , isppr   , isstgr , ilzr   , ihzr ,
     &                      wvmina , wvmaxa  ,
     &                      ita    , ida     ,
     &                      teta   , teda    ,
     &                      fpec   , fpec_max,
     &                      ncmt_stack       , cmt_stack
     &                    )
      implicit none
c-----------------------------------------------------------------------
c
c  ***************** fortran77 subroutine: xxdata_40 ******************
c
c  purpose:  To fetch  data  from  an input feature photon emissivity
c            file for a given emitting element superstage .
c
c  calling programs: adas416/dxdata_40
c
c  data:    Up to 'nstore' sets (data-blocks) of data may be  read from
c           the file - each block forming a complete feature photon
c           emissivity coefft. for given temp/density grid and wave-.
c           length range.  Each data-block  is  analysed independently
c           of any  other datablock.
c
c           the units used in the data file are taken as follows:
c
c           temperatures : ev
c           densities    : cm-3
c           pec          : phot. cm3 s-1 pixel-1
c
c  subroutine:
c
c  input : (i*4)  iunit    = unit to which input file is allocated.
c          (i*4)  dsname   = name of opened data set on iunit
c
c          (i*4)  nstore   = maximum number  of  input data-blocks  that
c                            can be stored.
c          (i*4)  npix    = maximum number  of  pixels in a data-blocks
c                            that can be stored.
c          (i*4)  ntdim    = max number of electron temperatures allowed
c          (i*4)  nddim    = max number of electron densities    allowed
c          (i*4)  ndptnl   = maximum level of partitions
```

```
c          (i*4)  ndptn    = maximum no. of partitions in one level
c          (i*4)  ndptnc   = maximum no. of components in a partition
c          (i*4)  ndcnct   = maximum number of elements in connection
c          (i*4)  ndstack  = maximum number of partition text lines
c          (i*4)  ndcmt    = maximum number of comment text lines
c                             vector
c output:  (i*4)  iz0      = read - emitting ion - nuclear charge
c          (i*4)  is       = read - emitting ion - charge
c                            (generalised to superstage label)
c          (i*4)  is1      = read - emitting ion - charge + 1
c                            (generalised to superstage index= is + 1)
c          (c*2)  esym     = read - emitting ion - element symbol
c
c          (i*4)  nptnl    = number of partition levels in block
c          (i*4)  nptn()   = number of partitions in partition level
c                            1st dim: partition level
c          (i*4)  nptnc(,) = number of components in partition
c                            1st dim: partition level
c                            2nd dim: member partition in partition level
c          (i*4)  iptnla() = partition level label (0=resolved root,1=
c                                                    unresolved root)
c                            1st dim: partition level index
c          (i*4)  iptna(,) = partition member label (labelling starts at 0)
c                            1st dim: partition level index
c                            2nd dim: member partition index in partition
c                            level
c          (i*4)  iptnca(,,)= component label (labelling starts at 0)
c                            1st dim: partition level index
c                            2nd dim: member partition index in partition
c                            level
c                            3rd dim: component index of member partition
c          (i*4)  ncnct     = number of elements in connection vector
c          (i*4)  icnctv()  = connection vector of number of partitions
c                             of each superstage in resolved case
c                             including the bare nucleus
c                             1st dim: connection vector index
c          (i*4)  ncptn_stack = number of text lines in partition block
c          (c*80) cptn_stack()= text lines in partition block
c                               1st dim: text line index (1->ncptn_stack)
c
c          (l*4)  lres      = .true.  => partial file
c                           = .false. => not partial file
c          (l*4)  lptn      = .true.  => partition block present
c                           = .false. => partition block not present
c          (l*4)  lcmt      = .true.  => comment text block present
c                           = .false. => comment text block not present
c          (l*4)  lsup      = .true.  => ss use of filmem field
c                           = .false. => old use of filmem field
c
c          (i*4)  nbsel     = number of data-blocks accepted & read in.
c          (i*4)  isela()   = read - data-set data-block entry indices
c                             dimension: data-block index
c
c          (i*4)  npixa()   = number of pixels for data block
c                             1st dim: data-block index
c          (c*8)  cfile()   = specific ion file source string in older
c                             forms.  Field not present in superstage
```

```
c                             version, but reused for added information
c                             1st dim: data-block index
c          (c*8)  ctype()  = data type string
c                             1st dim: data-block index
c          (c*2)  cindm()  = metastable index string
c                             1st dim: data-block index
c
c          (i*4)  isppr()   = parent index for each feature block
c                               1st dim: index of block in adf40 file
c          (i*4)  ispbr()   = base index for each feature block
c                               1st dim: index of block in adf40 file
c          (i*4)  isstgr()  = s1 for each resolved data block
c                               1st dim: index of block in adf40 file
c          (i*4)  ilzr()    = lowest ion charge relating to feature
c                               1st dim: index of block in adf40 file
c          (i*4)  ihzr()    = highest ion charge relating to feature
c                               1st dim: index of block in adf40 file
c
c          (r*8)  wvmina()  = lowest wavelength of feature block
c                               dimension: data-block index
c          (r*8)  wvmaxa()  = highest wavelength of feature block
c                               dimension: data-block index
c
c          (i*4)  ita()     = number of electron temperatures
c                               dimension: data-block index
c          (i*4)  ida()     = read - number of electron densities
c                               1st dim: data-block index
c
c          (r*8)  teta(,)   =  electron temperatures (units: ev)
c                               1st dim: electron temperature index
c                               2nd dim: data-block index
c          (r*8)  teda(,)   = electron densities (units: cm-3)
c                               1st dim: electron density index
c                               2nd dim: data-block index
c
c          (r*8)  fpec(,,,) = feature photon emissivity coeffts
c                               1st dim: pixel index
c                               2nd dim: electron temperature index
c                               3rd dim: electron density index
c                               4th dim: data-block index
c          (r*8)  fpec_max()= feature photon emissivity coefft. power
c                               integral maximum (over wavelength interval)
c                               as a function of Te at first Ne value
c                               1st dim: data-block index
c          (i*4)  ncmt_stack = number of text lines in comment block
c          (c*80) cmt_stack()= text lines in comment block
c                                1st dim: text line index (1->ncmt_stack)
c
c routine: (i*4)  i4eiz0   = function - (see routines section below)
c          (i*4)  i4fctn   = function - (see routines section below)
c          (i*4)  i4unit   = function - (see routines section below)
c          (i*4)  iblk     = array index: data-block index
c          (i*4)  itt      = array index: electron temperature index
c          (i*4)  itd      = array index: electron density     index
c          (i*4)  ntnum    = number of electron temperatures for current
c                            data-block
c          (i*4)  ndnum    = number of electron densities    for current
```

```
c                             data-block
c          (i*4)  iabt     = return code from 'i4fctn'
c          (i*4)  ipos1    = general use string index variable
c          (i*4)  ipos2    = general use string index variable
c
c          (l*4)  lbend    = identifies whether the last of the  input
c                            data sub-blocks has been located.
c                            (.true. => end of sub-blocks reached)
c
c          (c*1)  cslash   = '/' - delimiter for 'xxhkey'
c          (c*2)  c2       = general use two byte character string
c          (c*5)  ionnam   = emitting ion read from dataset
c          (c*6)  ckey1    = 'filmem' - input block header key
c          (c*4)  ckey2    = 'type  ' - input block header key
c          (c*4)  ckey3    = 'indm  ' - input block header key
c          (c*4)  ckey4    = 'isel  ' - input block header key
c          (c*80) c80      = general use 80 byte  character  string  for
c                            the input of data-set records.
c
c routines:
c          routine    source    brief description
c          ------------------------------------------------------------
c          i4eiz0     adas      returns z0 for given element symbol
c          i4fctn     adas      convert character string to integer
c          i4unit     adas      fetch unit number for output of messages
c          r8fctn     adas      convert string to real number
c          xxmkrp     adas      make up root partition text lines
c          xxcase     adas      convert a string to upper or lower case
c          xxhkey     adas      obtain key/response strings from text
c          xxrptn     adas      analyse an adf11 file partition block
c          xxword     adas      extract position of number in buffer
c          xxslen     adas      find string less front and tail blanks
c
c author:  h. p. summers, university of strathclyde
c          ja7.08
c          tel. 0141-548-4196
c
c date:    13/06/06
c
c
c version  : 1.1
c date     : 25-11-2004
c modified : martin o'mullane
c              - first version
c
c version  : 1.2
c date     : 29-11-2004
c modified : martin o'mullane
c              - faulty 1001 format statement.
c
c version  : 1.3
c date     : 15-05-2006
c modified : Hugh Summers
c              - complete rewrite for  operation with superstages and
c                partitions, made similar to xxdata_15.for .
c
c version  : 1.4
```

```
c date     : 06-11-2006
c modified : Allan Whiteford
c              - correction of indexing npixa by ipx rather than iblk.
c
c version  : 1.5
c date     : 15-01-2007
c modified : Hugh Summers
c              - corrected metastable count for Ne+0.
c
c-----------------------------------------------------------------------
      integer    idword    , idcnct  , iz0_max_res
c-----------------------------------------------------------------------
      parameter ( idword = 256 , idcnct = 100 )
      parameter ( iz0_max_res = 10 )
c-----------------------------------------------------------------------
      integer    i4eiz0    , i4fctn   , i4unit
      integer    iunit     , nstore   , ndpix         ,
     &           ntdim     , nddim    ,
     &           iz0       , is       ,
     &           is1       , nbsel
      integer    iblk      ,
     &           itt       , itd      ,
     &           ntnum     , ndnum    ,
     &           ipos1     , ipos2    , iabt
      integer    ndptnl    , ndptn    , ndptnc        , ndcnct
      integer    ndstack   , ndcmt
      integer    nptnl     , ncnct    , ncptn_stack   , ncmt_stack  ,
     &           iptnl
      integer    nfirst    , iwords   , nwords
      integer    i         , j        , ifirst        , ilast
      integer    max_indm  , indm     , ipx
c-----------------------------------------------------------------------
      real*8     fpmax     , r8fctn
c-----------------------------------------------------------------------
      logical    lbend
      logical    lptn      , lresol   , lres    , lsup
      logical    lcmt      , lptn_temp
c-----------------------------------------------------------------------
      character  dsname*80             , esym*2
      character  cslash*1              , colon*1
      character  c2*2                  , c3*3                ,
     &           ckey1*6               , ckey2*4             ,
     &           ckey3*4               , ckey4*4             ,
     &           ckey5*2               , ckey6*2             ,
     &           ckey7*2               , ckey8*2             ,
     &           ckey9*2               , ckey10*2            ,
     &           ckey11*4              , ckey12*4            ,
     &           ionnam*5              , c80*80              , cstrg*80
      character  cblnk8*8
c-----------------------------------------------------------------------
      integer    isela(nstore)         ,
     &           ita(nstore)           , ida(nstore)
      integer    nptn(ndptnl)          , nptnc(ndptnl,ndptn)
      integer    iptnla(ndptnl)        , iptna(ndptnl,ndptn)
      integer    iptnca(ndptnl,ndptn,ndptnc)
      integer    icnctv(ndcnct)
      integer    ifirsta(idword)       , ilasta(idword)
```

```
      integer    isstgr(nstore)        , ilzr(nstore)     , ihzr(nstore)
      integer    ispbr(nstore)         , isppr(nstore)
      integer    ncncta(iz0_max_res)   , icnctva(iz0_max_res,idcnct)
      integer    npixa(nstore)
c-----------------------------------------------------------------------
      character  cindm(nstore)*2       , cfile(nstore)*8      ,
     &           ctype(nstore)*8
      character  cptn_stack(ndstack)*80,cmt_stack(ndcmt)*80
c-----------------------------------------------------------------------
      real*8     teta(ntdim,nstore)    , teda(nddim,nstore)
      real*8     wvmina(nstore)        , wvmaxa(nstore)
      real*8     fpec(ndpix,ntdim,nddim,nstore)
      real*8     fpec_sum(ntdim,nddim,nstore)
      real*8     fpec_max(nstore)
c-----------------------------------------------------------------------
      save       cslash                ,
     &           ckey1                 , ckey2                ,
     &           ckey3                 , ckey4
c-----------------------------------------------------------------------
      data       cslash / '/' /        , colon  / ':' /
      data       cblnk8 /'        '/
      data       ckey1  / 'filmem' /   , ckey2  / 'type'   /  ,
     &           ckey3  / 'indm'   /   , ckey4  / 'isel'   /  ,
     &           ckey5  /  'pl'    /   , ckey6  /  'ss'    /  ,
     &           ckey7  /  'pb'   /    , ckey8  /  'pp'    /  ,
     &           ckey9  /  'lz'   /    , ckey10 /  'hz'   /   ,
     &           ckey11 / 'ispb'   /   , ckey12 / 'ispp'   /

      data  ncncta(1),(icnctva(1,i),i=1,2)   / 2,1,1/
      data  ncncta(2),(icnctva(2,i),i=1,3)   / 3,2,1,1/
      data  ncncta(3),(icnctva(3,i),i=1,4)   / 4,1,2,1,1/
      data  ncncta(4),(icnctva(4,i),i=1,5)   / 5,2,1,2,1,1/
      data  ncncta(5),(icnctva(5,i),i=1,6)   / 6,2,2,1,2,1,1/
      data  ncncta(6),(icnctva(6,i),i=1,7)   / 7,4,2,2,1,2,1,1/
      data  ncncta(7),(icnctva(7,i),i=1,8)   / 8,3,4,2,2,1,2,1,1/
      data  ncncta(8),(icnctva(8,i),i=1,9)   / 9,4,3,4,2,2,1,2,1,1/
      data  ncncta(9),(icnctva(9,i),i=1,10)  /10,2,4,3,4,2,2,1,2,1,1/
      data  ncncta(10),(icnctva(10,i),i=1,11)/11,2,2,4,3,4,2,2,1,2,1,1/
c-----------------------------------------------------------------------
      lbend = .false.
c-----------------------------------------------------------------------
```

Notes:

xxdata_46.for

```
      subroutine xxdata_46(iunit    ,
     &                     ndlev    , ndtrn     , ndmet     , ndprt    ,
     &                     ndomgl   , ndqd      , ndrep     , ndlrep   ,
     &                     ndte     , nddens    ,
     &                     seq      , iz0       , iz        , iz1      ,
     &                     ctype    , esym      ,
     &                     bwno_i   , nlvl_i    , lmet_i    , lcstrg_i ,
     &                     ia_i     , code_i    , cstrga_i  ,
     &                     isa_i    , ila_i     , xja_i     , wa_i     ,
     &                     nmet_i   , imeta_i   , nlexc_i   ,
     &                     bwno_f   , nlvl_f    , lmet_f    , lcstrg_f ,
     &                     ia_f     , code_f    , cstrga_f  ,
     &                     isa_f    , ila_f     , xja_f     , wa_f     ,
     &                     nmet_f   , imeta_f   , nlexc_f   ,
     &                     ntrn     , idxl      , idxu      , ityp     ,
     &                     aul      , nomgl     , omgl      ,
     &                     npol     , idxp      , dpol      ,
     &                     lmax_qd  , qd1       , qd2       ,
     &                     nl1      , nl2       , nl3       ,
     &                     inrep    , nrep      , ilrep     , lrep     ,
     &                     nte      , te        , ndens     , dens     ,
     &                     ntp      , tp        , ndensp    , densp    ,
     &                     zp       , amsp      ,
     &                     l_trn    , l_pol     , l_def     , l_rep    ,
     &                     l_plasma , l_te      , l_dens    , l_tp     ,
     &                     l_densp  , l_zp      , l_amsp
     &                    )

      implicit none
c-----------------------------------------------------------------------
c  ****************** fortran77 subroutine: xxdata_46 ******************
c
c  purpose:  to fetch data from an adf46 data set.
c
c  input : (i*4)  iunit       = unit to which input file is allocated
c          (i*4)  ndlev       = maximum number of energy levels in
c                               either ion stage
c          (i*4)  ndtrn       = maximum number of transitions
c          (i*4)  ndmet       = maximum number of metastables
c          (i*4)  ndprt       = maximum number of parents (except
c                               metastables)
c          (i*4)  ndomgl      = maximum number of partial waves
c          (i*4)  ndqd        = maximum number of quantum defect expansion
c          (i*4)  ndrep       = maximum number of representative
c                               n-shell
c          (i*4)  ndlrep      = maximum number of representative
c                               l-shells
c          (i*4)  ndte        = maximum number of temperatures
c          (i*4)  nddens      = maximum number of densities
c
c  output: (c*2)  seq         = iso-electronic sequence symbol
c          (i*4)  iz0         = nuclear charge
c          (i*4)  iz          = recombined ion charge
```

```
c          (i*4)  iz1         = recombining ion charge (=iz+1)
c          (c*2)  ctype       = adf46 file resol. ('ca', 'ls' or 'ic')
c          (r*8)  bwno_i      = ionis. potential of recombining ion (cm-1)
c          (i*4)  nlvl_i      = number of levels of recombining ion
c          (l*4)  lmet_i      = .true.  => recombined metastables marked
c                               .false. => recombining metastables unmarked
c                                          (default action - mark ground)
c          (l*4)  lcstrg_i    = .true.  => standard config strings for
c                                          recombining ion states
c                               .false. => unreadable config string for
c                                          at least one recombining ion state
c          (i*4)  ia_i()      = index of recombining ion levels
c                               1st dim: recombining ion level index
c          (c*1)  code_i()    = met. or excit. DR parent marker (* or #)
c                               1st dim: recombining ion level index
c          (i*(*))cstrga_i()  = ionised ion configuration strings
c                               1st dim: recombining ion level index
c          (i*4)  isa_i()     = ionised ion level multiplicity
c                               1st dim: recombining ion level index
c          (i*4)  ila_i()     = ionised ion total orb. ang. mom.
c                               1st dim: recombining ion level index
c          (r*8)  xja_i()     = recombining ion level (stat wt-1)/2
c                               1st dim: recombining ion level index
c          (r*8)  wa_i()      = recombining ion level wave number (cm-1)
c                               1st dim: recombining ion level index
c          (i*4)  nmet_i      = number of recombining ion metastables
c          (i*4)  imeta_i()   = pointers to recombining metastables in full
c                               recombined ion state list
c                               1st dim: recombining metastable index
c          (i*4)  nlexc_i     = number of recombining ion excited levels
c          (r*8)  bwno_f      = ionis. potential of recombined ion (cm-1)
c          (i*4)  nlvl_f      = number of levels of recombined ion (cm-1)
c          (l*4)  lmet_f      = .true.  => recombined metastables marked
c                               .false. => recombined metastables unmarked
c                                          (default action - mark ground)
c          (l*4)  lcstrg_f    = .true.  => standard config strings for
c                                          recombined ion states
c                               .false. => unreadable config string for
c                                          at least one recombined ion state
c          (i*4)  ia_f()      = index of recombined ion levels
c                               1st dim: recombined ion level index
c          (c*1)  code_f()    = met. or excit. DR parent marker (* or #)
c                               1st dim: recombined ion level index
c          (i*(*))cstrga_f()  = ionised ion configuration strings
c                               1st dim: recombined ion level index
c          (i*4)  isa_f()     = ionised ion level multiplicity
c                               1st dim: recombined ion level index
c          (i*4)  ila_f()     = ionised ion total orb. ang. mom.
c                               1st dim: recombined ion level index
c          (r*8)  xja_f()     = recombined ion level (stat wt-1)/2
c                               1st dim: recombined ion level index
c          (r*8)  wa_f()      = recombined ion level wave number (cm-1)
c                               1st dim: recombined ion level index
c          (i*4)  nmet_f      = number of recombined ion metastables
c          (i*4)  imeta_f()   = pointers to recombined metastables in full
c                               recombined ion state list
c                               1st dim: recombined metastable index
```

```
c          (i*4)  nlexc_f     = number of recombined ion excited levels
c          (i*4)  ntrn        = number of transitions
c          (i*4)  idxl()      = index of lower parent ion level of trans.
c                               1st dim: transition index
c          (i*4)  idxu()      = index of upper parent ion level of trans.
c                               1st dim: transition index
c          (i*4)  ityp()      = type of parent transition (1=dipol,
c                                  2=non-dipol, 3=spin change)
c                               1st dim: transition index
c          (r*8)  aul()       = transition probabilities (s-1)
c                               ist dim: transition index
c          (r*8)  nomgl()     = number threshold partial wave collision
c                               strengths for parent transitions
c                               1st dim: transition index
c          (r*8)  omgl(,)     = threshold partial wave collision
c                               strengths for parent transitions
c                               1st dim: partial wave index (=l+1)
c                               2nd dim: transition index
c          (i*4)  npol        = number of polarisabilies
c          (i*4)  idxp()      = index of parent ion level
c                               1st dim: polarisabilities index
c          (r*8)  dpol()      = dipole polarisability of parent level
c                               1st dim: polarisability index
c          (i*4)  lmax_qd     = largest quantum defect l-series expansion
c          (r*8)  qda1(,)     = quantum defect expansion coefficient a0
c                               1st dim: quantum defect l-series +1
c                               2nd dim: polarisability index
c          (r*8)  qda2(,)     = quantun defect expansion coefficient a1
c                               1st dim: quantum defect l-series +1
c                               2nd dim: polarisability index
c          (i*4)  nl1         = lowest n-shell of recombined ion for DR
c          (i*4)  nl2         = lowest l-resolved shell of recombined ion
c                               for DR
c          (i*4)  nl3         = highest bundle-n n-shell for recombined
c                               ion for DR
c          (i*4)  inrep       = number of representative n-shell
c          (i*4)  nrep()      = representative n-shells
c                               1st dim: representative n-shell index
c          (i*4)  ilrep       = number of representative l-shells
c          (i*4)  lrep()      = representative l-shells
c                               1st dim: representative l-shell index
c          (i*4)  nte         = number of electron temperatures
c          (r*8)  te()        = electron temperatures (K)
c                               1st dim: electron temperatures index
c          (i*4)  ndens       = number of electron densities
c          (r*8)  dens()      = electron densities (cm-3)
c                               1st dim: electron densities index
c          (i*4)  ntp         = number of positive ion temperatures
c          (r*8)  tp()        = positive ion temperatures (K)
c                               1st dim: ion temperatures index
c          (i*4)  ndensp      = number of positive ion densities
c          (r*8)  densp()     = positive ion densities (cm-3)
c                               1st dim: ion densities index
c          (i*4)  zp          = projectile ion effective charge
c          (i*4)  amsp        = projectile ion effective mass number
c          (l*4)  l_trn       = .true. => transition data present
c                               .false.=> transition data not present
```

```
c          (l*4)  l_pol       = .true. => polarisability data present
c                               .false.=> polarisability data not present
c          (l*4)  l_def       = .true. => quantum defect data present
c                               .false.=> quantum defect data not present
c          (l*4)  l_rep       = .true. => representative n data present
c                               .false.=> representative n data not present
c          (l*4)  l_plasma    = .true. => plasma data present
c                               .false.=> plasma data not present
c          (l*4)  l_te        = .true. => electron temperature data present
c                               .false.=> electron temperature data not present
c          (l*4)  l_dens      = .true. => electron density data present
c                               .false.=> electron density data not present
c          (l*4)  l_tp        = .true. => ion temperature data present
c                               .false.=> ion temperature data not present
c          (l*4)  l_densp     = .true. => ion density data present
c                               .false.=> ion density data not present
c          (l*4)  l_zp        = .true. => ion effective z data present
c                               .false.=> ion effective z data not present
c          (l*4)  l_amsp      = .true. => ion effective mass data present
c                               .false.=> ion effective mass data not present
c
c
c
c  routines:
c          routine    source    brief description
c          ------------------------------------------------------------
c          i4unit     adas      fetch unit number for output of messages
c          i4eiz0     adas      fetch nuclear charge for element symbol
c          xfesym     adas      fetch element symbol for nuclear charge
c          xxcase     adas      convert string to lower or upper case
c          xxword     adas      convert string to separate words
c          xxhkey     adas      extract a key name value from a string
c          xxlast     adas      find last occurence of char in string
c          xxslen     adas      find first and last characters of string
c          xxdtes     adas      detect if config string is eissner/standard
c          xxcftr     adas      covert config string between eissner/standard
c
c  author:  Alessandra Giunta
c  date  :  13-02-2008
c
c
c  version  : 1.1
c  date     : 13-02-2008
c  modified :
c               - first version
c
c  version  : 1.2
c  date     : 30-05-2008
c  modified : Hugh Summers, University of Strathclyde
c               - logic change for sequence symbol reading from input
c                 file.
c
c  version  : 1.3
c  date     : 13-02-2009
c  modified : Hugh Summers, University of Strathclyde
c               - changed to adf46 with substantial re-positioning
c                 and re-definition of fields.
```

```
c
c
c
c-----------------------------------------------------------------------
      integer   nsym     , ndword
c-----------------------------------------------------------------------
      parameter( nsym = 92 , ndword = 60 )
c-----------------------------------------------------------------------
      integer   i4unit     , i4eiz0     , lenstr
      integer   iunit      , icount
      integer   ind1       , ind2       , istart   , istop  ,
     &          i          , j          , indx_fb  , indx_lb
      integer   nte        , ntp        ,
     &          ndens      , ndensp     , nqd
      integer   ndlev      , ndtrn      , ndmet    , ndte   ,
     &          ndprt      , ndomgl     , ndrep    , ndlrep ,
     &          nddens     , ndqd
      integer   iz0        , iz         , iz1      , izp
      integer   nlvl_i     , nlvl_f
      integer   nfirst     , iwords     , nwords   , iword  ,
     &          ilen_index , ilen_cnfg  , ilen_s   ,
     &          ilen_l     , ilen_j     , ilen_wnf
      integer   nmet_i     , nlexc_i
      integer   nmet_f     , nlexc_f
      integer   nvlce      , ilen
      integer   ilexc_i    , ilexc_f    , ntrn
      integer   lmax_omgl  , lmax_qd    , npol
      integer   nl1        , nl2        , nl3      ,
     &          inrep      , ilrep
c-----------------------------------------------------------------------
      real*8    bwno_f       , bwno_i   , amsp     , zp
c-----------------------------------------------------------------------
      character cdelim*1
      character seq*2      , esym*2     , ctype*2  , xfesym*2
      character c80*80     , c10*10     , c22*22   , c256*256 ,
     &          c200*200
      character cline*80   , clong*256
      character cbreak*1   , csrch*1
      character f_1004*41  , f_1005*29  , f_1006*30
c-----------------------------------------------------------------------
      logical   lmet_f     , lcstrg_f ,
     &          lmet_i     , lcstrg_i
      logical   lstan      , leiss
      logical   l_trn      , l_pol      , l_def     , l_rep   ,
     &          l_plasma   , l_te       , l_dens    , l_tp    ,
     &          l_densp    , l_zp       , l_amsp
      logical   l_nl1      , l_nl2      , l_nl3     ,
     &          l_nrep     , l_lrep
c-----------------------------------------------------------------------
      integer   ia_i(ndlev)     , indpol(ndlev)
      integer   isa_i(ndlev)    , ila_i(ndlev)
      integer   ia_f(ndlev)
      integer   isa_f(ndlev)    , ila_f(ndlev)
      integer   ifirst(ndword)  , ilast(ndword)
      integer   imeta_i(ndmet)  , imeta_f(ndmet)
      integer   idxl(ndtrn)     , idxu(ndtrn)   , idxp(ndlev)
      integer   nrep(ndrep)     , lrep(ndlrep)  , nomgl(ndtrn)
```

```
      integer   indi(ndtrn)     , indf(ndtrn)   , ityp(ndtrn)
c-----------------------------------------------------------------------
      real*8    xja_i(ndlev)    , wa_i(ndlev)
      real*8    xja_f(ndlev)    , wa_f(ndlev)
      real*8    te(ndte)        , tp(ndte)
      real*8    dens(nddens)    , densp(nddens)
      real*8    aul(ndtrn)      , dpol(ndlev)
      real*8    omgl(ndomgl,ndtrn)
      real*8    qd1(ndqd,ndlev) , qd2(ndqd,ndlev)
c-----------------------------------------------------------------------
      character code_f(ndlev)*1 , cstrga_f(ndlev)*(*) ,
     &          code_i(ndlev)*1 , cstrga_i(ndlev)*(*)
      character ckey(3)*6       , cans(3)*4
c-----------------------------------------------------------------------
      data      ckey/'seq','nucchg','type'/
      data      cbreak,csrch/'/',')'/
c-----------------------------------------------------------------------
```

Notes:

xxdtes.for

```
      subroutine xxdtes( cstrg , leiss  , lstan  ,  nvlce )
c
      implicit none
c-----------------------------------------------------------------------
c
c  ****************** fortran77 subroutine: xxdtes *********************
c
c  purpose: Detects if the configuration string from a specific ion
c           level list line is of eissner form , standard form or
c           neither.
C
C           If neither, the subroutine checks to see if it is a
c           bundle (* in the string) or based on a parent ([..] in
c           the string).  If the string is of Eissner or standard
c           form, the n-shell and l-shell of the outermost
c           (valence)  electron is returned.
c
c           A version of this routine with a more extended return of
c           parameters and bale to handle very long configuration
c           strings is available as 'g5dtes.for'.
c
c  calling programs: general use
c
c  subroutine:
c
c  input : (c*(*)) cstrg    = configuration character string
c  output: (l*4)   leiss    = .true.  => eissner form
c                             .false. => not eissner form
c  output: (l*4)   lstan    = .true.  => standard form
c                             .false. => not standard form
c  output: (i*4)   nvlce    = outer electron n-shell if recognisable
c
c          (l*4)   lbndl    = .true.  => bundled form ('*' found)
c                             .false. => not bundled form
c          (l*4)   lprnt    = .true.  => parent form ('[...]' found)
c                             .false. => not parent form
c          (c*19)  cstr_top = leading part of config. string in Eissner
c                             format (no leading blank, trailing blanks)
c          (c*(*)) cstr_tail= trailing part of config. string in Eissner
c                             format (no leading blank, trailing blanks)
c          (i*4)   lvlce    = outer electron l-shell if recognisable
c
c          (i*4)   i       = general use
c          (i*4)   iabt    = return code (see specific function)
c                            0 => ok
c                            1 => fault detected
c          (i*4)   icfsel  = 1 => standard form out, standard form in
c                            2 => eissner  form out, standard form in
c                            3 => standard form out, eissner  form in
c                            4 => eissner  form out, eissner  form in
c          (i*4)   ishel   = shell counter
c          (i*4)   ip      = parity of configuration
c          (i*4)   maxn    = n_shell sum for configuration
```

```
c          (i*4)   nshel   = number of shells identified ffrom string
c          (i*4)   ndword  = maximum number of words in string
c          (i*4)   nfirst  = first word to be extracted from string
c          (i*4)   nwords  = number of words in string
c          (i*4)   nela()  = number of electrons in each shell
c          (i*4)   ifirst()= position of first char. of word in string
c          (i*4)   ilast() = position of last char. of word in string
c
c          (c*1)   cdelim  = separators for words in string
c          (c*19)  cstrgo  = general use string
c          (c*19)  strg    = standard form configuration string
c          (c*19)  strge   = eissner form configuration string
c          (c*1)   cheisa()= eissner character for orbitals
c          (c*2)   chstda()= standard orbital spec. for each shell
c          (c*2)   cnela() = chars. for no. of equiv. elec. in shell
c                            (eissner form case)
c          (c*1)   chqa()  = index to hexadecimal conversions
c          (c*1)   chra()  = char. for no. of. equiv. elec. in shell
c                            (standard form case)
c
c routines:
c          routine    source   brief description
c          ------------------------------------------------------------
c          i4fctn     adas     converts character string to integer
c          i4ngrp     adas     returns n quantum number in the
c                              eissner single hexadecimal character form
c          i4pgrp     adas     returns parity of orbital given the
c                              eissner single hexadecimal character form
c          i4schr     adas     returns numerical value for number of
c                              equivalent electrons given as hex> char.
c          cstgrp     adas     returns term of orbital given in the
c                              eissner single hexadecimal character form
c          ceigrp     adas     returns eissner code for orbital
c          xxword     adas     finds number of words in a string
c          xxcmps     adas     compare standard config. strings
c
c
c
c author:  h. p. summers, university of strathclyde
c          ja8.08
c          tel. 0141-553-4196
c
c date:    19/02/03
c
C VERSION: 1.1                          DATE: 19-1-96
C MODIFIED: TIM HAMMOND (TESSELLA SUPPORT SERVICES PLC)
C               - PUT UNDER S.C.C.S. CONTROL
C
C VERSION: 1.2                          DATE: 14-10-96
C MODIFIED: WILLIAM OSBORN (TESSELLA SUPPORT SERVICES PLC)
C               - ADDED CHANGES DATED 01/10/96 ABOVE
C
C VERSION: 1.3                          DATE: 28-08-97
C MODIFIED: HUGH SUMMERS
C               - ADDED CHANGES TO CHECK 'G' STATES
C
C VERSION: 1.4                         DATE: 19/02/03
```

```
C MODIFIED: HUGH SUMMERS
C               - Rewrite based on g5dtes.for
C
C VERSION: 1.5                          DATE: 28/09/2004
C MODIFIED: Martin O'Mullane
C               - Incorrect redirection when checking the Eissner pattern.
C                 The if statement block checking ir jumped out of the
C                 current sub-block to the end of the previous sub-block
C                 rather than to the end of its own sub-block.
C
C VERSION: 1.6                          DATE: 17/05/2007
C MODIFIED: Allan Whiteford
C           - Updated comments as part of subroutine documentation
C             procedure.
C
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      integer   ndword
c-----------------------------------------------------------------------
      parameter ( ndword = 21 )
c-----------------------------------------------------------------------
      integer   nvlce     , lvlce
      integer   i         , nfirst    , iwords  , nwords
      integer   i4unit
      integer   i4ngrp    , i4lgrp    , i4pgrp  , i4schr   , i4fctn
      integer   ishel     , nshel     , ip      , maxn     , icfsel
      integer   iabt      , i1        , i2      , i3       , ir
      integer   lenstr    , lenwrd    , nw4
c-----------------------------------------------------------------------
      character cstrg*(*) , cstr_top*19 , cstr_tail*40   , fmt*12
      character cstrgo*19 , strg*24   , strge*19
      character cstr19*19 , cdelim*1
      character cstgrp*2  , ceigrp*1   , cstr2*2     , cstr1*1
c-----------------------------------------------------------------------
      logical*4 leiss     , lstan     , lequiv
      logical*4 lbndl     , lprnt
c-----------------------------------------------------------------------
      integer   nela(ndword)   , ifirst(ndword)      , ilast(ndword)
c-----------------------------------------------------------------------
      character cheisa(ndword)*1 , chstda(ndword)*2  , cnela(ndword)*2
      character chqa(61)*1  , chra(ndword)*1
C-----------------------------------------------------------------------
      data  chqa    /'1','2','3','4','5','6','7','8','9',
     &               'A','B','C','D','E','F','G','H','I',
     &               'J','K','L','M','N','O','P','Q','R',
     &               'S','T','U','V','W','X','Y','Z','a',
     &               'b','c','d','e','f','g','h','i','j',
     &               'k','l','m','n','o','p','q','r','s',
     &               't','u','v','w','x','y','z'/
      data  cdelim  /' '/
c-----------------------------------------------------------------------
```

Notes:

xxcftr.for

```
      SUBROUTINE XXCFTR( ICFSEL , CSTRGI , CSTRGO )
C
      IMPLICIT NONE
C-----------------------------------------------------------------------
C
C  ****************** FORTRAN77 SUBROUTINE: XXCFTR *********************
C
C  PURPOSE: CONVERTS A CONFIGURATION CHARACTER STRING, SUCH AS OCCURS
C           IN A SPECIFIC ION FILE LEVEL LIST, BETWEEN EISSNER AND
C           STANDARD FORMS
C
C  CALLING PROGRAMS: GENERAL USE
C
C  SUBROUTINE:
C
C  INPUT : (I*4)   ICFSEL  = 1 => STANDARD FORM OUT, STANDARD FORM IN
C                            2 => EISSNER  FORM OUT, STANDARD FORM IN
C                            3 => STANDARD FORM OUT, EISSNER  FORM IN
C                            4 => EISSNER  FORM OUT, EISSNER  FORM IN
C  INPUT : (C*(*)) CSTRGI  = CONFIGURATION STRING IN INPUT FORM
C  OUTPUT: (C*(*)) CSTRGO  = CONFIGURATION STRING IN OUTPUT FORM
C
C          (I*4)   I       = GENERAL USE
C          (I*4)   ISHEL   = SHELL COUNTER
C          (I*4)   IP      = PARITY OF CONFIGURATION
C          (I*4)   MAXN    = N_SHELL SUM FOR CONFIGURATION
C          (I*4)   NSHEL   = NUMBER OF SHELLS IDENTIFIED fFROM STRING
C          (I*4)   ISTART  = START OF STRING EXCLUDING INITIAL BLANKS
C          (I*4)   ISTOP   = END OF STRING EXCLUDING FINAL BLANKS
C          (I*4)   IREM    = REMAINDER OF EISSNER STRING MOD 3
C          (I*4)   NELA()  = NUMBER OF ELECTRONS IN EACH SHELL
C
C          (C*19)  STRG    = STANDARD FORM CONFIGURATION STRING
C          (C*19)  STRGE   = EISSNER FORM CONFIGURATION STRING
C          (C*1)   CHEISA()= EISSNER CHARACTER FOR ORBITALS
C          (C*2)   CHSTDA()= STANDARD ORBITAL SPEC. FOR EACH SHELL
C                            (EISSNER FORM CASE)
C          (C*1)   CHQA()  = INDEX TO HEXADECIMAL CONVERSIONS
C          (C*1)   CHRA()  = CHAR. FOR NO. OF. EQUIV. ELEC. IN SHELL
C                            (STANDARD FORM CASE)
C
C          (L*4)   LEISS   = .TRUE.  => EISSNER FORM
C                            .FALSE. => NOT EISSNER FORM
C
C
C ROUTINES:
C          ROUTINE    SOURCE   BRIEF DESCRIPTION
C          ------------------------------------------------------------
C          I4UNIT     ADAS     FETCH UNIT NUMBER FOR OUTPUT OF MESSAGES
C          I4NGRP     ADAS     RETURNS N QUANTUM NUMBER IN THE
C                              EISSNER SINGLE HEXADECIMAL CHARACTER FORM
C          I4PGRP     ADAS     RETURNS PARITY OF ORBITAL GIVEN THE
C                              EISSNER SINGLE HEXADECIMAL CHARACTER FORM
C          I4SCHR     ADAS     RETURNS NUMERICAL VALUE FOR NUMBER OF
C                              EQUIVALENT ELECTRONS GIVEN AS HEX> CHAR.
C          CSTGRP     ADAS     RETURNS TERM OF ORBITAL GIVEN IN THE
```

```
C                               EISSNER SINGLE HEXADECIMAL CHARACTER FORM
C           CEIGRP      ADAS    RETURNS EISSNER CODE FOR ORBITAL
C           XXSLEN      ADAS     FINDS STRING LENGTH EXCLUDING LEADING AND
C                                TRAILING BLANKS
C
C
C NOTE:     THE ROUTINE IS USED TO CONVERT THE CONFIGURATION CHARACTER
C           STRING OCCURRING IN ADF04 FILE LEVEL LISTS.  THE STRING
C           LENGTH ALLOCATED TO THIS IS *18 FOLLOWING 1 BLANK SPACE
C           AFTER THE LEVEL INDEX.  A PROBLEM ARISES WHEN THE FIRST
C           SHELL CONTAINS MORE THAN 9 EQUIVALENT ELECTRONS.  IN THIS
C           CASE, OVERSPILL IS ALLOWED INTO THE BLANK CHARACTER SPACE.
C           THE ROUTINE WILL ANALYSE A *19 STRING INCLUDING THE USUALLY
C           BLANK LOCATION OR A *18 STRING EXCLUDING IT.  IN THE LATTER
C           CASE AN INTELLIGENT GUESS IS MADE AS TO WHETHER THE OMITTED
C           BLANK SHOULD IN FACT BE A '1'.  THIS SITUATION OCCURS FOR A
C           LEADING CLOSED D-SHELL.
C
C AUTHOR:   H. P. SUMMERS, UNIVERSITY OF STRATHCLYDE
C           JA8.08
C           TEL. 0141-553-4196
C
C DATE:     25/10/95
C
C
C UPDATE:   19/02/03  H. P. SUMMERS - EXTENDED RANGE AND STRINGS
C
C
C UPDATE:   09/05/04  H. P. SUMMERS - CATCH FULL EISSNER FORM (IE. +50)
C                                     FOR 1ST SHELL DISPLACED TO COL. 2
C                                     IN 19 CHAR STRING
C
C
C UNIX-IDL PORT:
C
C VERSION: 1.1                          DATE: 19-1-96
C MODIFIED: TIM HAMMOND (TESSELLA SUPPORT SERVICES PLC)
C               - PUT UNDER SCCS CONTROL
C
C VERSION: 1.2                          DATE: 19-02-03
C MODIFIED: H. P. SUMMERS
C - EXTENDED RANGE AND STRINGS
C
C VERSION: 1.3                          DATE: 09-05-04
C MODIFIED: H. P. SUMMERS
C - CATCH EISSNER FULL STRING FORM PROBLEM
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      INTEGER   ICFSEL    , I
      INTEGER   I4UNIT    , I4NGRP    , I4PGRP  , I4SCHR
      INTEGER   ISHEL     , NSHEL     , IP      , MAXN
      INTEGER   IABT      , ISTART    , ISTOP   , IREM
C-----------------------------------------------------------------------
      CHARACTER CSTRGI*(*), CSTRGO*(*) , CSTR19*25
      CHARACTER STRG*19   , STRGE*19
      CHARACTER CSTGRP*2  , CEIGRP*1
```

```
C-----------------------------------------------------------------------
      LOGICAL*4 LEISS
C-----------------------------------------------------------------------
      INTEGER   NELA(6)
C-----------------------------------------------------------------------
      CHARACTER CHEISA(6)*1 , CHSTDA(6)*2
      CHARACTER CHQA(61)*1  , CHRA(6)*1
C-----------------------------------------------------------------------
      DATA CHQA     /'1','2','3','4','5','6','7','8','9',
     &               'A','B','C','D','E','F','G','H','I',
     &               'J','K','L','M','N','O','P','Q','R',
     &               'S','T','U','V','W','X','Y','Z','a',
     &               'b','c','d','e','f','g','h','i','j',
     &               'k','l','m','n','o','p','q','r','s',
     &               't','u','v','w','x','y','z'/
C-----------------------------------------------------------------------
```

Notes:

g5dtes.for

```
      subroutine g5dtes( cstrg , leiss  , lstan  , lbndl  , lprnt ,
     &                   cstr_top       , cstr_tail       ,
     &                   nvlce , lvlce
     &                 )
c
      implicit none
c-----------------------------------------------------------------------
c
c  ****************** fortran77 subroutine: g5dtes **********************
c
c  purpose: detects if the configuration string from a specific ion
c           level list line is of eissner form , standard form or
c           neither.  If neither, the subroutine checks to see if it
c           is a bundle (* in the string) or based on a parent ([..]
c           in the string).  If the string is of Eissner or standard
c           form, the n-shell and l-shell of the outermost (valence)
c           electron is returned.
c
c  calling programs: general use
c
c  subroutine:
c
c  input : (c*(*)) cstrg    = configuration character string
c  output: (l*4)   leiss    = .true.  => eissner form
c                             .false. => not eissner form
c  output: (l*4)   lstan    = .true.  => standard form
c                             .false. => not standard form
c  output: (l*4)   lbndl    = .true.  => bundled form ('*' found)
c                             .false. => not bundled form
c  output: (l*4)   lprnt    = .true.  => parent form ('[...]' found)
c                             .false. => not parent form
c  output: (c*19)  cstr_top = leading part of config. string in Eissner
```

```
c                              format (no leading blank, trailing blanks)
c  output: (c*(*)) cstr_tail= trailing part of config. string in Eissner
c                              format (no leading blank, trailing blanks)
c  output: (i*4)   nvlce    = outer electron n-shell if recognisable
c  output: (i*4)   lvlce    = outer electron l-shell if recognisable
c
c          (i*4)   i       = general use
c          (i*4)   iabt    = return code (see specific function)
c                            0 => ok
c                            1 => fault detected
c          (i*4)   icfsel  = 1 => standard form out, standard form in
c                            2 => eissner  form out, standard form in
c                            3 => standard form out, eissner  form in
c                            4 => eissner  form out, eissner  form in
c          (i*4)   ishel   = shell counter
c          (i*4)   ip      = parity of configuration
c          (i*4)   maxn    = n_shell sum for configuration
c          (i*4)   nshel   = number of shells identified ffrom string
c          (i*4)   ndword  = maximum number of words in string
c          (i*4)   nfirst  = first word to be extracted from string
c          (i*4)   nwords  = number of words in string
c          (i*4)   nela()  = number of electrons in each shell
c          (i*4)   ifirst()= position of first char. of word in string
c          (i*4)   ilast() = position of last char. of word in string
c
c          (c*1)   cdelim  = separators for words in string
c          (c*19)  cstrgo  = general use string
c          (c*19)  strg    = standard form configuration string
c          (c*19)  strge   = eissner form configuration string
c          (c*1)   cheisa()= eissner character for orbitals
c          (c*2)   chstda()= standard orbital spec. for each shell
c          (c*2)   cnela() = chars. for no. of equiv. elec. in shell
c                            (eissner form case)
c          (c*1)   chqa()  = index to hexadecimal conversions
c          (c*1)   chra()  = char. for no. of. equiv. elec. in shell
c                            (standard form case)
c
c routines:
c          routine    source   brief description
c          ------------------------------------------------------------
c          i4fctn     adas     converts character string to integer
c          i4ngrp     adas     returns n quantum number in the
c                              eissner single hexadecimal character form
c          i4pgrp     adas     returns parity of orbital given the
c                              eissner single hexadecimal character form
c          i4schr     adas     returns numerical value for number of
c                              equivalent electrons given as hex> char.
c          cstgrp     adas     returns term of orbital given in the
c                              eissner single hexadecimal character form
c          ceigrp     adas     returns eissner code for orbital
c          xxword     adas     finds number of words in a string
c          xxcmps     adas     compare standard config. strings
c
c
c
c author:  h. p. summers, university of strathclyde
c          ja8.08
```

```
c          tel. 0141-553-4196
c
c date:    11/09/01
c
c update: 28/09/2004 Martin O'Mullane
C               - Incorrect redirection when checking the Eissner pattern.
C                 The if statement block checking ir jumped out of the
C                 current sub-block to the end of the previous sub-block
C                 rather than to the end of its own sub-block.
c
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      integer   ndword
c-----------------------------------------------------------------------
      parameter ( ndword = 21 )
c-----------------------------------------------------------------------
      integer   nvlce     , lvlce
      integer   i         , nfirst    , iwords  , nwords
      integer   i4unit
      integer   i4ngrp    , i4lgrp    , i4pgrp  , i4schr   , i4fctn
      integer   ishel     , nshel     , ip      , maxn     , icfsel
      integer   iabt      , i1        , i2      , i3       , ir
      integer   lenstr    , lenwrd    , nw4
c-----------------------------------------------------------------------
      character cstrg*(*) , cstr_top*19 , cstr_tail*(*)   , fmt*12
      character cstrgo*19 , strg*24   , strge*19
      character cstr19*19 , cdelim*1
      character cstgrp*2  , ceigrp*1   , cstr2*2     , cstr1*1
c-----------------------------------------------------------------------
      logical*4 leiss     , lstan     , lequiv
      logical*4 lbndl     , lprnt
c-----------------------------------------------------------------------
      integer   nela(ndword)   , ifirst(ndword)      , ilast(ndword)
c-----------------------------------------------------------------------
      character cheisa(ndword)*1 , chstda(ndword)*2  , cnela(ndword)*2
      character chqa(61)*1  , chra(ndword)*1
C-----------------------------------------------------------------------
      data  chqa    /'1','2','3','4','5','6','7','8','9',
     &               'A','B','C','D','E','F','G','H','I',
     &               'J','K','L','M','N','O','P','Q','R',
     &               'S','T','U','V','W','X','Y','Z','a',
     &               'b','c','d','e','f','g','h','i','j',
     &               'k','l','m','n','o','p','q','r','s',
     &               't','u','v','w','x','y','z'/
      data  cdelim  /' '/
c-----------------------------------------------------------------------
```

Notes:

xxdrbf.for

```
       subroutine xxdrbf( iz1    , eij    , fij   ,
     &                    ndte   , nte    , tea   ,
     &                    adgf
```

```
     &                 )
       implicit none
c-----------------------------------------------------------------------
c
c  ***************** fortran77 subroutine: xxdrbf.for *********************
c
c  purpose: to evaluate dielectronic recombination coefficients
c           using the Burgess General Formula in its original
c           form.
c
c  calling program: various
c
c  subroutine:
c
c  input : (i*4)  iz1      = recombining ion charge
c  input : (r*8)  eij      = z-scaled parent transition energy (Rydberg)
c                            (note absolute energy =(iz1+1)**2*eij)
c  input : (r*8)  fij      = parent transition oscillator strength
c  input : (i*4)  ndte     = maximum no. of electron temperatures
c  input : (i*4)  nte      = number of electron temperatures
c  input : (r*8)  tea()    = electron temperatures(Kelvin)
c                            1st dim: electron temperature index
c  output: (r*8)  adgf()   = zero density dielectronic recombination rate
c                            coefft. (cm3 s-1)
c                            1st dim: electron temperature index
c
c
c  routines:
c          none
c
c
c  author:  Alessandra Giunta, University of Strathclyde
c  date:    25-01-2008
c
c
c  version  : 1.1
c  date     : 25-01-2008
c  modified : Alessandra Giunta
c              - first version.
c
c-----------------------------------------------------------------------------
       integer   i4unit
       integer   ndte  , nte   , it
       integer   iz1
c-----------------------------------------------------------------------
       real*8    eij   , fij
       real*8    z1    , zz1   , z2   , zz2
       real*8    a     , b     , sa   ,
     &           x     , y
c-----------------------------------------------------------------------
       real*8    tea(ndte)     , adgf(ndte)
c-----------------------------------------------------------------------
```

Notes:

xxdrbp.for

```
       subroutine xxdrbp( ndpwv    , ndcor    , ndte     ,
     &                    iz0      , iz1      ,
     &                    ep       , fp       , wp_tr    , tr_ev  ,
     &                    jcor     , cor      ,
     &                    n        , defn     ,
     &                    nte      , tea_ev   ,
     &                    adrn_red , frac_nl  ,
     &                    adrn     , adrn_ps  , adrn_pi
     &                  )
       implicit none
c-----------------------------------------------------------------------
c
c  ***************** fortran77 subroutine: xxdrbp.for *********************
c
c  purpose: to evaluate n-shell and nl-shell selective dielectronic
c           recombination using the Burgess General Program in its
c           original form.
c
c  calling program: various
c
c  subroutine:
c
c  input : (i*4)  ndpwv    = maximum no. of partial waves (l-subshells)
c  input : (i*4)  ndcor    = maximum no. of Bethe correction factors
c  input : (i*4)  ndte     = maximum no. of electron temperatures
c  input : (i*4)  iz0      = nuclear charge
c  input : (i*4)  iz1      = recombining ion charge
c  input : (r*8)  ep       = parent transition energy (cm-1)
c  input : (r*8)  fp       = parent transition oscillator strength
c  input : (r*8)  wp_tr    = parent trans. external radiation field
c                            dilution factor
c  input : (r*8)  tr_ev    = external radiation field Planckian
c                            temperature (eV)
c  input : (i*4)  jcor     = number of Bethe correction factors
c  input : (r*8)  cor()    = Bethe correction factors
c                            1st dim: partial wave index (=l+1)
c  input : (i*4)  n        = principal quantum shell
c  input : (r*8)  defn     = mean quantum defect for n-shell
c  input : (i*4)  nte      = number of electron temperatures
c  input : (r*8)  tea_ev() = electron temperatures (eV)
c                            1st dim: electron temperature index
c
c  output: (r*8)  adrn_red = reduced dielectronic recombination coefft. to
c                            n-shell
c  output: (r*8)  frac_nl()= frac. of n-shell recomb. to selected l-shell
c                            1st dim: l-shell index (=l+1)
c  output: (r*8)  adrn()   = dielectronic recombination rate coefft. to
c                            n-shell (cm3 s-1)
c                            1st dim: electron temperature index
c  output: (r*8)  adrn_ps()= stimulated dielectronic recombination rate
c                            coefft. to n-shell
c                            1st dim: electron temperature index
c  output: (r*8)  adrn_pi()= dielectronic photo-ionisation rate
c                            coefft. from n-shell via resonance state
```

```
c                           1st dim: electron temperature index
c
c
c  routines:
c          ------------------------------------------------------------
c          xxbfme    adas      bound-free hydrogenic radial integrals
c          i4unit    adas      fetch unit number for output of messages
c
c
c  author:  H. P. Summers, University of Strathclyde
c           tel: 0141-548-4196
c
c  date:    01/11/07
c
c
c  version  : 1.1
c  date     : 29-05-2009
c  modified : H P Summers
c               - first version.
c
c
c-----------------------------------------------------------------------
       integer   idpwv
c-----------------------------------------------------------------------
       real*8    ryd_wno  , ev_kel  , ryd_ev   , boltz_kel
       real*8    zero     , zero_e  , zero_d
c-----------------------------------------------------------------------
       parameter ( idpwv  = 1000      , ryd_wno = 109727.26d0 )
       parameter ( ev_kel = 11605.4d0 , ryd_ev  = 13.6048d0   )
       parameter ( boltz_kel = 157890.0d0 )
       parameter ( zero = 1.0d-72 , zero_e = -1.65d2 , zero_d = -7.2d1 )
c-----------------------------------------------------------------------
       integer   i4unit
       integer   ndpwv  , ndcor   , ndte
       integer   iz0    , iz1     , jcor
       integer   n
       integer   nte
       integer   j1     , j       ,it
c-----------------------------------------------------------------------
       real*8    ep     , fp      , wp_tr    , tr_ev  , defn
       real*8    adrn_red
       real*8    z      , eij     , f        , t      , def    ,
     &           ad     , zz      , z1       , en     , t1     ,
     &           z2     , e       , a       , b       ,
     &           tj     , th      , t3       , c3     ,
     &           x      , ate     , fac      , wep_tr
c-----------------------------------------------------------------------
       real*8    cor(ndcor)
       real*8    theta(idpwv)
       real*8    frac_nl(ndpwv)
       real*8    tea_ev(ndte)
       real*8    adrn(ndte)     , adrn_ps(ndte) , adrn_pi(ndte)
c-----------------------------------------------------------------------
```

Notes:

gxdrbp.for

```
       subroutine gxdrbp( ndrep     , ndte     , ndcor     ,
     &                    iz0       , iz1      ,
     &                    ep        , fp       , np        , lp      ,
     &                    ng        , lg       ,
     &                    jcor      , cor      , df        ,
     &                    nmin      , def_nmin , phfrac    , corfac  ,
     &                    inrep     , nrep     ,
     &                    nte       , tea_ev   , dens    ,
     &                    adrn_rep  , adr_tot  ,
     &                    lrep      , lcut     , lcor
     &                  )

       implicit none
c-----------------------------------------------------------------------
c
c  ****************** fortran77 subroutine: gxdrbp *********************
c
c purpose : to evaluate the total dielectronic rate coefficient at a
c           set of electron temperatures using the Burgess General
c           Program. A finite electron density reduction of the total
c           coefft. may be applied. The partial rate coefficient to a
c           representative of n-shells is also evaluated.
c
c notes:    low partial wave Bethe collision strength correction factors
c           are applied according to the parent transition type as:
c
c       type  transition       (cor(j),j=1,jcor)              df
c           ----  ------------------  ----------------------------  ---
c        1    ng=1,np>=2,lp=lg+1: 0.05,0.30,0.50,0.90      2.0
c        2    ng=2,np=3 ,lp=lg+1: 0.01,0.02,0.20,0.40,0.70,0.90  1.0
c        3    ng=2,np=3,lp=lg-1 : 0.01,0.01,0.01,0.08,0.30,0.70  1.0
c        4    np-ng=0, lp=lg+1  : 0.30,0.35,0.40,0.45,0.70,0.90  0.5
c        5    np-ng=0, lp=lg-1  : 0.30,0.35,0.40,0.45,0.70,0.90  0.5
c        6    np-ng>0, lp=lg+1  : 0.01,0.02,0.20,0.40,0.70,0.90  1.0
c        7    np-ng>0, lp=lg-1  : 0.01,0.01,0.01,0.08,0.30,0.70  1.0
c
c            Results are adjustable via two global parameters, phfrac and
c            corfac which adjust the phase space availability of the lowest
c            accessible n-shell and modify the Bethe corrections
c            respectively.  phfrac is a simple multiplier.  corfac applies
c            as:
c                   cor_new(j)=exp(-corfac/((j-1)**df+0.5))*cor(j)
c
c            where j-1 = l (the partial wave).
c
c  calling program: various
c
c
c
c  subroutine:
c
c  input : (i*4)  ndrep    = maximum no. of representative n-shells
```

```
c  input : (i*4)  ndcor    = maximum no. of Bethe correction factors
c  input : (i*4)  ndte     = maximum no. of electron temperatures
c  input : (i*4)  iz0      = nuclear charge
c  input : (i*4)  iz1      = recombining ion charge
c  input : (r*8)  ep       = parent transition energy (cm-1)
c  input : (r*8)  fp       = parent transition oscillator strength
c  input : (i*4)  np       = active electron n-shell for upper level
c                            of parent transition
c  input : (i*4)  lp       = active electron l-shell for upper level
c                            of parent transition
c  input : (i*4)  ng       = active electron n-shell for ground level
c                            of parent transition
c  input : (i*4)  lg       = active electron l-shell for ground level
c                            of parent transition
c  input : (i*4)  jcor     = number of partial wave corrections
c  input : (i*4)  cor()    = partial wave corrections
c  input : (i*4)  df       = partial wave scaling parameter
c                            1st dim: partial wave index (=l+1)
c  input : (i*4)  nmin     = lowest n-shell of recombined system
c  input : (r*8)  def_nmin = quantum defect applicable to lowest
c                            n-shell for recombination
c  input : (r*8)  phfrac   = phase space occupancy availability
c                            for lowest accessible n-shell
c  input : (r*8)  corfac   = global adjustment for bethe corrections
c                            in general program
c  input : (i*4)  inrep    = number of representative levels
c  input : (i*4)  nrep()   = representative n-shells
c                            1st dim: representative n-shell index
c  input : (i*4)  nte      = number of electron temperatatures
c  input : (r*8)  tea_ev() = electron temperatures (eV)
c                            1st dim: electron temperature index
c  input : (r*8)  dens     = electron density (cm-3)
c
c
c  output: (r*8)  adrn_rep(,) = partial dielectronic coeffts. for
c                               representative n-shells (cm^3 s^-1).
c                               1st dim: representative n-shell index
c                               2nd dim: electron temperature index
c  output: (r*8)  adr_tot()   = total dielectronic coeffts (cm^3 s^-1.
c                               1st dim: electron temperature index
c  output: (l*4)  lrep        = .true  => input represent. levels used
c                               .false.=> default internal values used
c  output: (l*4)  lcut        = .true  => input density used for cutoff
c                               .false.=> default internal values used
c  output: (l*4)  lcor        = .true  => input cor values used
c                               .false.=> default internal values used
c
c  routines:
c          routine   source    description
c          ---------------------------------------------------
c          xxdrbp    adas      Burgess program DR coefft. to n-shell
c          xxbfme    adas      bound-free hydrogenic radial integrals
c          i4unit    adas      fetch unit number for output of messages
c
c
c  author:  H. P. Summers, University of Strathclyde
c           tel: 01235-46-4459
```

```
c
c  date:   29/05/2009
c
c
c  version  : 1.1
c  date     : 29-05-2009
c  modified : H P Summers
c              - first version.
c
c-----------------------------------------------------------------------
      integer     idrep      , idte       , idcor      , ndpwv
c-----------------------------------------------------------------------
      real*8      ryd_wno   , ev_kel      , ryd_ev     , boltz_kel
c-----------------------------------------------------------------------
      parameter ( idrep = 30 , idte = 20  , idcor = 20  , ndpwv = 1000 )
      parameter ( ev_kel = 11605.4d0   , ryd_ev  = 13.6048d0   )
      parameter ( boltz_kel = 157890.0d0  , ryd_wno = 109727.26d0 )
c-----------------------------------------------------------------------
      integer     i4unit
      integer     ndrep  , ndte   , ndcor
      integer     iz0    , iz1    ,
     &            np     , lp     , ng     , lg     ,
     &            nmin   , inrep  , nte    ,
     &            jcor_int
      integer     inrep_def       , inrep_int        , nmin_int ,
     &            nmax_int
      integer     i      , i0     , i1     , iopt    ,
     &            j      , jcor   , n      , n1     , it
c-----------------------------------------------------------------------
      real*8      a      , b       , ate     ,
     &            defn   , defn1   , df      ,
     &            v      , v1      ,
     &            xl     , z1      , zz1     ,
     &            vcut
      real*8      ep    , fp      , df_int    ,
     &            def_nmin , phfrac  , corfac  ,
     &            dens
      real*8      wp       , wp_tr   , tr_ev     ,
     &            adrn_red , adrn1_red
c-----------------------------------------------------------------------
      logical     lrep   , lcut      , lcor
c-----------------------------------------------------------------------
      integer     nrep_def(idrep), nrep_int(idrep), nrep(ndrep)
      integer     ncut(idte)
      integer     jcora(7)
c-----------------------------------------------------------------------
      real*8     cora(idcor,7)         , dfa(7)         , cor(ndcor)
      real*8     tea_ev(ndte)          , cor_int(idcor)
      real*8     adrn(idte)            , adrn_ps(idte)  , adrn_pi(idte)
      real*8     adrn1(idte)           , adrn1_ps(idte) ,adrn1_pi(idte)
      real*8     adrn_rep(ndrep,ndte)  , adr_tot(ndte)
      real*8     frac_nl(ndpwv)
c-----------------------------------------------------------------------
      data jcora/20,20,20,20,20,20,20/
      data dfa/2.0d0,1.0d0,1.0d0,0.5d0,0.5d0,1.0d0,1.0d0/
      data (cora(j,1),j=1,20)/0.05d0,0.3d0,0.5d0,0.9d0,16*1.0d0/
      data (cora(j,2),j=1,20)/0.01d0,0.02d0,0.20d0,0.40d0,0.70d0,0.90d0,
```

```
     &14*1.0d0/
      data (cora(j,3),j=1,20)/0.01d0,0.01d0,0.01d0,0.08d0,0.30d0,0.70d0,
     &14*1.0d0/
      data (cora(j,4),j=1,20)/0.30d0,0.35d0,0.40d0,0.45d0,0.70d0,0.90d0,
     &14*1.0d0/
      data (cora(j,5),j=1,20)/0.30d0,0.35d0,0.40d0,0.45d0,0.70d0,0.90d0,
     &14*1.0d0/
      data (cora(j,6),j=1,20)/0.01d0,0.02d0,0.20d0,0.40d0,0.70d0,0.90d0,
     &14*1.0d0/
      data (cora(j,7),j=1,20)/0.01d0,0.01d0,0.01d0,0.08d0,0.30d0,0.70d0,
     &14*1.0d0/
      data inrep_def /    30/
      data nrep_def  /     1,    2,    3,    4,   5,   6,   7,   8,   9,
     &                    10,   12,   15,   20,  30,  40,  50,  60,  70,
     &                   100,  150,  200,  250, 300, 400, 500, 600, 700,
     &                   800,  900, 1000/
c-----------------------------------------------------------------------
```

Notes:

Appendix D

Shell scripts

Scripts **adas1#1**	Current location	Central ADAS	
		CVS	Rel
nist_get_data.pl	/home/hps/adas_dev/offline_adas/adas1#1/scripts/	n	n
process_nist_to_adf00.pl	/home/hps/adas_dev/offline_adas/adas1#1/scripts/	n	n
process_nist_to_adf04.pl	/home/hps/adas_dev/offline_adas/adas1#1/scripts/	n	n

Scripts **adas7#3**	Current location	Central ADAS	
		CVS	Rel
setup_isoseq_pwb_adf27.pl	/home/hps/adas_dev/offline_adas/adas7#3/scripts/	n	n
setup_isoseq_dw_adf27.pl	/home/hps/adas_dev/offline_adas/adas7#3/scripts/	n	n
setup_isoseq_dw_bbgp_adf27	/home/hps/adas_dev/offline_adas/adas7#3/scripts/	n	n
process_ion_pwb_adf27_to_adf04.pl	/home/hps/adas_dev/offline_adas/adas7#3/scripts/	n	n
process_ion_dw_adf27_to_adf04.pl	/home/hps/adas_dev/offline_adas/adas7#3/scripts/	n	n
process_ion_dw_bbgp_adf27_to_adf04.pl	/home/hps/adas_dev/offline_adas/adas7#3/scripts/	n	n
adas7#3_pwb_llbatch.pl	/home/hps/adas_dev/offline_adas/adas7#3/scripts/	n	n
adas7#3_dw_llbatch.pl	/home/hps/adas_dev/offline_adas/adas7#3/scripts/	n	n
adas7#3_dw_bbgp_llbatch.pl	/home/hps/adas_dev/offline_adas/adas7#3/scripts/	n	n

Script	Current location	Local checks			Central ADAS	
		Txt	Opr	Lnk	CVS	Rel
run_archive_808_scripts.pl	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
run_archive_808_paper.pl	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
run_808_offline.sh	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
run_opt_808_offline.sh	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
run_813_offline.sh	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n
generate_adf11_classes_10-12.pl	/home/hps/adas_dev/idl/adaslib/read_adf/	y	n	n	n	n

nist_get_data.pl

```
#! /usr/bin/perl -w

#------------------------------------------------------------------------
#
# PROJECT:
#       ADAS
#
# NAME:
#       nist_get_data.pl
#
# PURPOSE:
#       Extract energy level information from NIST for an element.  The energy
#       level page for each ion of an element is obtained from the web, stripped
#       of redundant material and archived in an element directory.
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        nistroot     =    user root directory to nist energy level archive
#                          (typically '/home/<userid>/nist_energy_level_tables/')
#        userpostfix  =    personal name postfix for use in archived data sets
#                          (as '/nistroot//<ion>_<userpostfix>.dat')
#        element      =    element full name
#
# NOTES:
#
#         Archives energy level tables from NIST as
#               ~/nistroot/<element name>/<elem.sym.><ion charge>_<userpostfix>.dat
#
#         A typical command line call is:
#              nist_get_data.pl --nistroot=/home/hps/nist_energy_level_tables/
#                               --userpostfix=hps --element=radon
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   30/07/2012
#
#------------------------------------------------------------------------
```

process_nist_to_adf00.pl

```
#! /usr/bin/perl -w

#-----------------------------------------------------------------------------
#
# PROJECT:
#       ADAS
#
# NAME:
#       process_nist_to_adf00.pl
#
# PURPOSE:
#       Ionisation limit and ground state quantum number data are extracted from the ADAS
#       NIST data archive, assembled and written to adf00.  The 2012 format adf00 with outer
#       quantum numbers of ground levels is used.  Missing NIST data for an ion is substituted
#       with the 'T. H. Carlson, C. W. Nestor, N. Wassermann & J. D. McDowell (1970) Atomic
#       Data, 2, 63-99' values from the pre 2012 adf00 data collection, with outer quantum
#       numbers drawn from the standard (high z) list
#      (see ~/adas/adf00/adf00_extension_ground_level_list.dat).
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        nistroot     =    user root directory to nist energy level archive
#                          (typically '/home/<userid>/nist_energy_level_tables/')
#        userpostfix  =    personal name postfix for use in archived/archiving data sets
#                          (as '/nistroot/<ion>_<userpostfix>.dat')
#        element      =    element full name
#
# NOTES:
#
#         Procures energy level tables from NIST as
#               ~/nistroot/<element name>/<elem.sym.><ion charge>_<userpostfix>.dat
#         Archives adf00 dataset as
#               ~/adas_dev/adas/adf00/<element name>/<elem.sym.><ion charge>_<userpostfix>.dat
#
#         A typical command line call is:
#              process_nist_to_adf00.pl --nistroot=/home/hps/nist_energy_level_tables/
#                                       --userpostfix=hps --element=radon
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   13/07/2012
#
#-----------------------------------------------------------------------------
```

process_nist_to_adf04.pl

```
#! /usr/bin/perl -w

#-----------------------------------------------------------------------
#
# PROJECT:
#       ADAS
#
# NAME:
#       process_nist_to_adf04.pl
#
# PURPOSE:
#
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        nistroot     =    user root directory to nist energy level archive
#                          (typically '/home/<userid>/nist_energy_level_tables/')
#        userpostfix  =    personal name postfix for use in archived/archiving data sets
#                          (as '/nistroot/<ion>_<userpostfix>.dat')
#        element      =    element full name
#
# NOTES:
#
#         Processes energy level tables extracted and archived from NIST as
#               ~/nistroot/<element name>/<elem.sym.><ion charge>_<userpostfix>.dat
#         Extracts quantum numbers and energy levels.  Archives as adf04 format dataset
#         energy level lists, but with no transition lines as
#               ~/adas_dev/adas/adf04/nist#<nuclear charge>/<ion>.dat
#
#         A typical command line call is:
#              process_nist_to_adf04.pl --nistroot=/home/hps/nist_energy_level_tables/
#                                       --userpostfix=hps --element=radon
#
# WARNING:
#         NIST data can have a configuration string with an unspecified n-shell (eg. Al0: 'nd').
#         Such energy level lines must be deleted from the NIST dataset before processing.
#         The program accepts '?' queried values in NIST data but removes the '?'.
#         Only the LS coupling scheme is recognised.  Other schemes have the 2S+1 and L fields
#         set to blank in the output adf04 dataset.
#         Repeated J-values for a single energy level in NIST data (eg. '(1/2,3/2,5/2)') give
#         an entry in the output adf04 datset for the last J-value only.
#         Parentage information in the NIST configuration field is removed.
#         Energy level displacement information in the NIST data (eg. '+x') is removed.
#         The ionisation potential in the output adf04 dataset is taken from adf00.  It is
#         assumed that ionisation potential extraction from NIST for adf00 update has been done
#         previously.
#
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
```

```
#
# VERSION:
#         1.1   30/07/2012
#
#-------------------------------------------------------------------------------
```

setup_isoseq_pwb_adf27.pl

```
#! /usr/bin/perl -w

#------------------------------------------------------------------------------
#
# PROJECT:
#       ADAS
#
# NAME:
#       setup_isoseq_pwb_adf27.pl
#
# PURPOSE:
#       Read /pwb/ adf27 template to set up /dw/ adf27 drivers for each ion
#       of an iso-electronic sequence.
#
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        adasroot     =    user root directory to adas code directories
#                          (typically '/home/<userid>/adas_dev')
#        userdircode  =    personal name prefix for adf27 and adf04 files
#                          (such as 'cophps' for hps produced datasets
#        isoseq       =    iso-electronic sequence symbol
#
# NOTES:
#        The standard file directory structure for adf27 is assumed
#         adf27:  <adasroot>/adas/adf27/pwb/<isoseq>like/<userdircode>#<isoseq>
#                       with member template
#         with targets
#
#         adf27:  <adasroot>/adas/adf27/pwb/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#
#         A typical command line call is:
#              setup_isoseq_pwb_adf27.pl --adasroot=/home/hps/adas_dev --userdircode=cophps
#                                        --isoseq=al
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   26/10/2011
#
#------------------------------------------------------------------------------
```

setup_isoseq_dw_adf27.pl

```
#! /usr/bin/perl -w

#------------------------------------------------------------------------------
#
```

```
# PROJECT:
#       ADAS
#
# NAME:
#       setup_isoseq_dw_adf27.pl
#
# PURPOSE:
#       Read /dw/ adf27 template and associated /pwb/ adf04 datasets produced
#       by Autostructure to set up /dw/ adf27 drivers for each ion
#       of an iso-electronic sequence present in the /pwb/ archive.
#
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        adasroot     =    user root directory to adas code directories
#                          (typically '/home/<userid>/adas_dev')
#        userdircode  =    personal name prefix for adf27 and adf04 files
#                          (such as 'cophps' for hps produced datasets
#        isoseq       =    iso-electronic sequence symbol
#
# NOTES:
#        The standard file directory structure for adf27 and adf04 is assumed
#         adf27:  <adasroot>/adas/adf27/pwb/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#         adf27:  <adasroot>/adas/adf27/dw/<isoseq>like/<userdircode>#<isoseq>
#                       with member template
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/pwb/
#                       with members /ic_<ion>.dat, /ic_<ion>t1.dat>,
#                                    /ls_<ion>.dat, /ls_<ion>t1.dat>,
#         with targets
#
#         adf27:  <adasroot>/adas/adf27/dw/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/dw/
#                       with members /ic_<ion>.dat, /ic_<ion>_t5.dat>,
#                                    /ls_<ion>.dat, /ls_<ion>_t5.dat>,
#
#         A typical command line call is:
#              setup_isoseq_dw_adf27.pl --adasroot=/home/hps/adas_dev --userdircode=cophps
#                                       --isoseq=al
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   16/12/2011
#
#-----------------------------------------------------------------------------
```

setup_isoseq_dw_bbgp_adf27.pl

```
#! /usr/bin/perl -w

#------------------------------------------------------------------------------
#
# PROJECT:
#       ADAS
#
# NAME:
#       setup_isoseq_dw_bbgp_adf27.pl
#
# PURPOSE:
#       Read /dw_bbgp/ adf27 template and associated /pwb/ adf04 datasets produced
#       by Autostructure to set up /dw_bbgp/ adf27 drivers for each ion
#       of an iso-electronic sequence present in the /pwb/ archive.
#
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        adasroot     =    user root directory to adas code directories
#                          (typically '/home/<userid>/adas_dev')
#        userdircode  =    personal name prefix for adf27 and adf04 files
#                          (such as 'cophps' for hps produced datasets
#        isoseq       =    iso-electronic sequence symbol
#
# NOTES:
#        The standard file directory structure for adf27 and adf04 is assumed
#         adf27:  <adasroot>/adas/adf27/pwb/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#         adf27:  <adasroot>/adas/adf27/dw_bbgp/<isoseq>like/<userdircode>#<isoseq>
#                       with member template
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/pwb/
#                       with members /ic_<ion>.dat, /ic_<ion>t1.dat>,
#                                    /ls_<ion>.dat, /ls_<ion>t1.dat>,
#         with targets
#
#         adf27:  <adasroot>/adas/adf27/dw_bbgp/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/dw_bbgp/
#                       with members /ic_<ion>.dat, /ic_<ion>_t6.dat>,
#                                    /ls_<ion>.dat, /ls_<ion>_t6.dat>,
#
#         A typical command line call is:
#              setup_isoseq_dw_bbgp_adf27.pl --adasroot=/home/hps/adas_dev
#                                            --userdircode=cophps --isoseq=al
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   19/09/2011
#
#------------------------------------------------------------------------------
```

process_ion_pwb_adf27_to_adf04.pl

```
#! /usr/bin/perl -w

#-----------------------------------------------------------------------------
#
# PROJECT:
#       ADAS
#
# NAME:
#       process_ion_pwb_adf27_to_adf04.pl
#
# PURPOSE:
#       Script for single ion execution of Autostructure for preparation of
#       plane wave Born (pwb) approximations of adf04 datasets in ls and ic
#       resolution and in types 1 and 3 from an adf27 driver. Can be used
#       directly or as the executable in loadleveler batch processing types 1
#       and 3.
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        adasroot     =    user root directory to adas code directories
#                          (typically '/home/<userid>/adas_dev')
#        userdircode  =    personal name prefix for adf27 and adf04 files
#                          (such as 'cophps' for hps produced datasets
#        isoseq       =    iso-electronic sequence symbol
#        ion          =    specific series member (eg ar5 equivalent to Arˆ+5)
#
# NOTES:
#        The standard file directory structure for adf27 and adf04 is assumned
#         adf27:  <adasroot>/adas/adf27/pwb/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/pwb/
#                       with members /ic_<ion>.dat, /ic_<ion>t1.dat>,
#                                    /ls_<ion>.dat, /ls_<ion>t1.dat>,
#
#         A typical command line call is:
#              process_ion_pwb_adf27_to_adf04.pl --adasroot=/home/hps/adas_dev
#                                                --userdircode=cophps --isoseq=al --ion=s3
#
#
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   12/09/2011
#
#-----------------------------------------------------------------------------
```

process_ion_dw_adf27_to_adf04.pl

```
#! /usr/bin/perl -w

#------------------------------------------------------------------------------
#
# PROJECT:
#       ADAS
#
# NAME:
#       process_ion_dw_adf27_to_adf04.pl
#
# PURPOSE:
#       Script for single ion execution of Autostructure for preparation of
#       distorted wave (dw) approximations of adf04 datasets in ls and ic
#       resolution and in types 5 and 3 from an adf27 driver. Can be used
#       directly or as the executable in loadleveler batch processing types 5
#       and 3.
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        adasroot     =    user root directory to adas code directories
#                          (typically '/home/<userid>/adas_dev')
#        userdircode  =    personal name prefix for adf27 and adf04 files
#                          (such as 'cophps' for hps produced datasets
#        isoseq       =    iso-electronic sequence symbol
#        ion          =    specific series member (eg ar5 equivalent to Arˆ+5)
#
# NOTES:
#        The standard file directory structure for adf27 and adf04 is assumned
#         adf27:  <adasroot>/adas/adf27/dw/<isoseq>like/<userdircode>#<isoseq>
#                        with members ic_<ion>.dat and ls_<ion>.dat
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/dw/
#                        with members /ic_<ion>.dat, /ic_<ion>t5.dat>,
#                                     /ls_<ion>.dat, /ls_<ion>t5.dat>,
#
#         A typical command line call is:
#              process_ion_dw_adf27_to_adf04.pl --adasroot=/home/hps/adas_dev
#                                               --userdircode=cophps --isoseq=al --ion=s3
#
#
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   12/09/2011
#
#------------------------------------------------------------------------------
```

process_ion_dw_bbgp_adf27_to_adf04.pl

```
#! /usr/bin/perl -w
```

```
#-----------------------------------------------------------------------------
#
# PROJECT:
#       ADAS
#
# NAME:
#       process_ion_dw_bbgp_adf27_to_adf04.pl
#
# PURPOSE:
#       Script for single ion execution of Autostructure for preparation of
#       distorted wave (dw_bbgp) approximations of adf04 datasets in ls and ic
#       resolution and in type 6 from an adf27 driver. Can be used
#       directly or as the executable in loadleveler batch processing type 6.
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        adasroot     =    user root directory to adas code directories
#                          (typically '/home/<userid>/adas_dev')
#        userdircode  =    personal name prefix for adf27 and adf04 files
#                          (such as 'cophps' for hps produced datasets
#        isoseq       =    iso-electronic sequence symbol
#        ion          =    specific series member (eg ar5 equivalent to Arˆ+5)
#
# NOTES:
#        The standard file directory structure for adf27 and adf04 is assumned
#         adf27:  <adasroot>/adas/adf27/dw_bbgp/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/dw_bbgp/
#                       with members /ic_<ion>.dat, /ic_<ion>t6.dat>,
#                                    /ls_<ion>.dat, /ls_<ion>t6.dat>,
#
#         A typical command line call is:
#              process_ion_dw_bbgp_adf27_to_adf04.pl --adasroot=/home/hps/adas_dev
#                                                    --userdircode=cophps --isoseq=al
#                                                    --ion=s3
#
#
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   19/09/2011
#
#-----------------------------------------------------------------------------
```

adas7#3_pwb_llbatch.pl

```
#! /usr/bin/perl -w

#-----------------------------------------------------------------------------
#
```

```
# PROJECT:
#       ADAS
#
# NAME:
#       adas7#3_pwb_batch.pl
#
# PURPOSE:
#       Batch execution of Autostructure for preparation of plane wave Born
#       (pwb) approximations of adf04 datasets in ls and ic resolution and in
#       types 1 and 3.
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        adasroot     =    user root directory to adas code directories
#                          (typically '/home/<userid>/adas_dev')
#        userdircode  =    personal name prefix for adf27 and adf04 files
#                          (such as 'cophps' for hps produced datasets
#        isoseq       =    iso-electronic sequence symbol
#
# NOTES:
#        The standard file directory structure for adf27 and adf04 is assumned
#         adf27:  <adasroot>/adas/adf27/pwb/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/pwb
#                       with members /ic_<ion>.dat, /ic_<ion>t1.dat>,
#                                   /ls_<ion>.dat, /ls_<ion>t1.dat>,
#
#         A typical command line call is:
#              adas7#3_pwb_llbatch.pl --adasroot=/home/hps/adas_dev --userdircode=cophps
#                                     --isoseq=al
#
#         Batch runs are set up under loadleveler with email with notification to user
#         <userid> and loadleveler output and error data to
#              <adasroot>/adas.pass/adasa7#3_pwb_llbatch_<ion>.out
#              <adasroot>/adas.pass/adasa7#3_pwb_llbatch_<ion>.err
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   12/09/2011
#
#-----------------------------------------------------------------------------
```

adas7#3_dw_llbatch.pl

```
#! /usr/bin/perl -w

#-----------------------------------------------------------------------------
#
# PROJECT:
#       ADAS
```

```
#
# NAME:
#       adas7#3_dw_batch.pl
#
# PURPOSE:
#       Batch execution of Autostructure for preparation of distorted wave
#       (dw) approximations of adf04 datasets in ls and ic resolution and in
#       types 5 and 3.
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        adasroot     =    user root directory to adas code directories
#                          (typically '/home/<userid>/adas_dev')
#        userdircode  =    personal name prefix for adf27 and adf04 files
#                          (such as 'cophps' for hps produced datasets
#        isoseq       =    iso-electronic sequence symbol
#
# NOTES:
#        The standard file directory structure for adf27 and adf04 is assumned
#         adf27:  <adasroot>/adas/adf27/dw/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/dw
#                       with members /ic_<ion>.dat, /ic_<ion>t5.dat>,
#                                    /ls_<ion>.dat, /ls_<ion>t5.dat>,
#
#         A typical command line call is:
#              adas7#3_dw_llbatch.pl --adasroot=/home/hps/adas_dev --userdircode=cophps
#                                    --isoseq=al
#
#         Batch runs are set up under loadleveler with email with notification to user
#         <userid> and loadleveler output and error data to
#              <adasroot>/adas.pass/adasa7#3_dw_llbatch_<ion>.out
#              <adasroot>/adas.pass/adasa7#3_dw_llbatch_<ion>.err
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   19/09/2011
#
#-----------------------------------------------------------------------------
```

adas7#3_dw_bbgp_llbatch.pl

```
#! /usr/bin/perl -w

#-----------------------------------------------------------------------------
#
# PROJECT:
#       ADAS
#
# NAME:
```

```
#       adas7#3_dw_bbgp_batch.pl
#
# PURPOSE:
#       Batch execution of Autostructure for preparation of distorted wave
#       (dw) approximations of adf04 datasets in ls and ic resolution
#       and in types 6.
#
# PARAMETERS:
#        tempdir      =    temporary directory for calculation
#        adasroot     =    user root directory to adas code directories
#                          (typically '/home/<userid>/adas_dev')
#        userdircode  =    personal name prefix for adf27 and adf04 files
#                          (such as 'cophps' for hps produced datasets
#        isoseq       =    iso-electronic sequence symbol
#
# NOTES:
#        The standard file directory structure for adf27 and adf04 is assumned
#         adf27:  <adasroot>/adas/adf27/dw_bbgp/<isoseq>like/<userdircode>#<isoseq>
#                       with members ic_<ion>.dat and ls_<ion>.dat
#         adf04:  <adasroot>/adas/adf04/<userdircode>#<isoseq>/dw_bbgp
#                       with members /ic_<ion>.dat, /ic_<ion>t6.dat>,
#                                    /ls_<ion>.dat, /ls_<ion>t6.dat>,
#
#         A typical command line call is:
#              adas7#3_dw_bbgp_llbatch.pl --adasroot=/home/hps/adas_dev
#                                         --userdircode=cophps --isoseq=al
#
#         Batch runs are set up under loadleveler with email with notification to user
#         <userid> and loadleveler output and error data to
#              <adasroot>/adas.pass/adasa7#3_dw_bbgp_llbatch_<ion>.out
#              <adasroot>/adas.pass/adasa7#3_dw_bbgp_llbatch_<ion>.err
#
# WRITTEN:
#       Hugh Summers, University of Strathclyde
#
# MODIFIED:
#       1.1     Hugh Summers
#                - First release
#
# VERSION:
#         1.1   19/09/2011
#
#-----------------------------------------------------------------------------
```

Index