libreadapec: Access to APEC Outputs

for version 0.3.0, 31 March 2015

Adam R. Foster (afoster@cfa.harvard.edu)

This manual is for the APEC Output File Access Library (libreadapec) (version 0.3.0, 31 March 2015),

Table of Contents

1 Background

This library provides routines to take the outputs of an APEC run, which contain the emissivity of elements from Hydrogen to Nickel in a hot collisional plamsa, and calculates the spectrum implied from these.

This library is part of the AtomDB project `www.atomdb.org`, and is based upon the Astrophysical Plasma Emission Code (APEC) and the accompanying atomic data (APED).

APEC outputs consist of 3 relevant files. The `line.fits` file contains the emissvities and wavelengths of each line, identified by element, ion and upper and lower levels of the transition. This is straightforward to interpret. By default, any lines with emissivity greater that 10^{-20} photons cm^3 s^{-1} will be included in this list.

The other two files contain the continuum and the pseudo continuum. By default, the apec code produces the continuum summed for each element assuming equilibrium at the temperature, density and time for the plasma - e.g. all 11 stages of Nitrogen are summed together with the appropriate fractions and stored as 1 continuum. Data such as this is stored in the `coco.fits` (coco = compressed continuum).

If the continuum is desired by ion, then a separate file of type `comp.fits` will be required. This is fundamentally the same as the `coco.fits` file, it just has the data additionally separated by ion.

The continuum data itself are split into two continuum components. The true continuum is formed by bremstrahlung, radiative recombination and two photon emission. The pseduo continuum is the sum of all lines which have emissivity $< 10^{-20}$ photons cm^3 s^{-1}. Both continua are stored as functions of Energy (in keV) and the continuum at this point, in photons cm^3 s^{-1} keV^{-1}. These points are calculated so that a linear interpolation between all of them will be accurate to within 1\% of the original continuum value.

All of the emissivities returned by this data have had an elemental abundance applied to them, i.e. the emissivity of each line has already bee multiplied by the element's relative abundance compared to hydrogen. By default in APEC, these abundances are taken from Table 2 of *Anders, E. and Grevesse, N., Geochimica et Cosmochimica Acta (ISSN 0016-7037), vol. 53, Jan. 1989, p. 197-214*, which are abundances for the solar photosphere. Below they are listed on a log scale, so the relative abundance of He to H is $10^{10.99}/10^{12.0} = 0.098$. When abundance can be varied, they are varied relative to this number, so supplying an abundance of He = 2.0 to one of the library routines actually means its abundance relative to H is twice the solar value, or 0.196.

`H`	12.0
`He`	10.99
`Li`	1.16
`Be`	1.15
`B`	2.6
`C`	8.56
`N`	8.05
`O`	8.93

F	4.56
Ne	8.09
Na	6.33
Mg	7.58
Al	6.47
Si	7.55
P	5.45
S	7.21
Cl	5.5
Ar	6.56
K	5.12
Ca	6.36
Sc	3.1
Ti	4.99
V	4.0
Cr	5.67
Mn	5.39
Fe	7.67
Co	4.92
Ni	6.25

2 Installation

Code Prerequisites:

- `libcfitsio` Allows reading in of FITS files, i.e. the AtomDB database.
- The AtomDB database, from www.atomdb.org.
- For test routines: the environment variable `ATOMDB` set to the directory containing the AtomDB database.

Beyond that, the standard `./configure; make; make install` procedure should work to install libareadapec. Note that both shared libraries and statically linked libraries will be generated.

2.1 Make Check

The `make check` command should run as you would expect, and hopefully passes all tests. Ensure that the `ATOMDB` environment variable is correctly set before running the tests otherwise they will fail.

3 Using the Library

`Libreadapec` is designed to return a spectrum for use in further processing by whatever software the user desires. As a result, there are several levels at which it can be used. The most basic use is simply to read in the fits files into structures, which the user can then process separately. The most useful feature is the ability to return spectra, either whole spectra, separated by element/ion, and with user specified elemental abundances.

The full interface for each function in `libreadapec` is described in Chapter 4 [LibraryContents], page 5. The library contains several overlapping functions, which give different levels of access to the calculations: basic use can be a single call to a wrapper program to get the ionization balance, while "power users" can play with individual routines to speed up the calculations which must be made frequently.

Remember that for the library to function, a copy of the AtomDB database must be installed and the environment variable `ATOMDB` must be set to point to it.

NOTE: At various points in the code, references are made to the HDU. These are the HDUs in the fits files, with each representing a new temperature/time/density combination. In the code here, these are numbered from 1, with 1 being the first HDU to contain the emissivity data. Note that in most cases, this will be actually be the third HDU, which is labelled 2 by most fits viewers, and 1 by libreadapec.

3.1 Basic Usage

To initially load in the all of the emissivity data, call `readapec_getdata`. This will provide an `EMISSION` structure which has all of the data from the fits files.

If you simply want a spectrum in one of the HDUs, `readapec_simple_spectrum` will provide this. You supply the edges of the energy bins, and the number of bins, and the spectrum is calculated for them and returned to the output vector spectrum.

If you want spectra for indivdual elements or ions, use `readapec_calc_hdu_elem_spectrum` or `readapec_calc_hdu_ion_spectrum` respectively.

If you do not know which HDU you are interested in, use `readapec_find_kT_hdu` to find the HDU with the nearest temperature (by deafult, all APEC outputs are at density of $1\mathrm{cm}^{-3}$, and equilibirum, so only temperature search function is provided for now.).

`Readapec_calc_ion_spectrum` is similar to `readapec_calc_hdu_elem_spectrum` or `readapec_calc_hdu_ion_spectrum`, except you sepecify the temperature, not the HDU block. This will also interpolate the spectra between the 2 blocks, to give the best estimate of the emissivities in between. If you wish to obtain the spectrum for all ions of an element, set rmJ=0.

3.2 Advanced Usage

Often it is desired not just to get the spectrum, but to change the abundances of the element in the spectrum. This is provided by the combination of `readapec_calc_allion_spectrum`, which returns an `EMISSION_LIST` structure, separating each spectrum by element or ion depending on the value of `byion`. By then passing this to `readapec_calc_total_emission_abundance` along with a list of the elements and the abundances, you can obtain a spectrum with the varied abundances.

4 Library Contents

The library contains many functions, as well as several fuctions which call many of the other functions sequentially. This enables either fine grained and more computationally efficient access for high power use, or for the user to simply ask for the ionization balance in a single call.

4.1 External Library Usage

It is expected that these codes will be used in a range of other models and systems. To aid in this we have included a sample PYTHON script in the python directory, which uses the CTYPES interface to talk to the library.

While there are many functions which can be called in this way, it is antipated that the spectrum for each ion should be calculated separately and multiplied by the relevant abundance. The wrapper describes the initialization and then calc_ion_spectrum routines.

For memory management purposes, it is possible to read the emissivity data into a global variable only once, then make repeated calls to this. The routine `readapec_getdata_extern` allows this. Once it is read in, it will be automatically picked up by other routines, and the `apec_data` variable should be set to `NULL`. This is the method used by, for example, the attached python library.

4.2 Library Functions

4.2.1 readapec_getdata

```
void readapec_getdata(char *linefile,
                              char *cocofile,
                              struct EMISSION **apec_data);
```

```
    /* This is just a wrapper for read_fits_spectrum, which reads in
     * all of the apec outputs
     *
     * INPUTS
     * linefile     filename of _line.fits apec file
     * cocofile     filename of _comp.fits or _coco.fits apec continuum
     *              file
     *
     * OUTPUTS
     * apec_data    all of the emissivity data
     */
```

4.2.2 readapec_getdata_extern

```
void readapec_getdata_extern(char *linefile,
                                     char *cocofile);

    /* This is just a wrapper for read_fits_spectrum_extrnal. It reads in
     * all the data from the emissivity files and puts it into a global
     * variable,apec_data_memstore
     *
     * INPUTS
     * linefile     filename of _line.fits apec file
     * cocofile     filename of _comp.fits or _coco.fits apec continuum
     *              file
     *
     * OUTPUTS
     * none (though apec_data_memstore now contains emissivity data)
     *
     */
```

4.2.3 readapec_simple_spectrum

```
void readapec_simple_spectrum(struct EMISSION *apec_data,
                              int nbins,
                              double *ebins,
                              int hdu,
                              double *spectrum);
    /* This routine calculates the spectrum for a specific HDU.
     * It does not separate the emissivity by ion or element,
     * and assumes an abundance of 1.0 for all elements.
     * It is the quick & dirty version
     *
     * INPUTS
     * apec_data    all of the emissivity data (returned by
     *              read_apec_outputs)
     * nbins        the number of bins the output spectrum
     * *ebins       the edges of the bins, in keV. Note that the array
     *              length is nbins+1. Must be monotonically increasing.
     * hdu          Which HDU in the fits file we are interested. Note
     *              that they are indexed from 1, with 1 being the first
     *              "EMISSIVITY" HDU. This is typically HDU 2 in most
     *              FITS files.
```

```
     *
     * OUTPUTS
     * *spectrum    The emissivity in ph cm3 s-1 bin-1
     */
```

4.2.4 readapec_calc_hdu_elem_spectrum

```
void readapec_calc_hdu_elem_spectrum(struct EMISSION *apec_data,
                            int nbins,
                            double *ebins,
                            int hdu,
                            int Z,
                            double *spectrum);
```

```
    /* This routine calculates the spectrum of one element for a specific
     * HDU. It assumes an abundance of 1.0 solar for all elements.
     * It is the quick & dirty version
     *
     * INPUTS
     * apec_data    all of the emissivity data (returned by
     *              read_apec_outputs)
     * nbins        the number of bins the output spectrum
     * *ebins       the edges of the bins, in keV. Note that the array
     *              length is nbins+1. Must be monotonically increasing.
     * hdu          Which HDU in the fits file we are interested. Note
     *              that they are indexed from 1, with 1 being the first
     *              "EMISSIVITY" HDU. This is typically HDU 2 in most
     *              FITS files.
     * Z            Atomic number of element of interest (e.g. 6=carbon)
     *
     * OUTPUTS
     * *spectrum    The emissivity in ph cm3 s-1 bin-1
     */
```

4.2.5 readapec_calc_hdu_ion_spectrum_part

```
void readapec_calc_hdu_ion_spectrum_part(struct EMISSION *apec_data,
                            int nbins,
                            double *ebins,
                            int hdu,
                            int Z,
                            int rmJ,
                            int doline,
                            int docont,
                            double *spectrum);

    /* This routine calculates the spectrum of an ion for a specific HDU.
     * It assumes an abundance of 1.0 for all elements.
     *
     * INPUTS
     * apec_data    all of the emissivity data (returned by
     *              read_apec_outputs)
     * nbins        the number of bins the output spectrum
     * *ebins       the edges of the bins, in keV. Note that the array
     *              length is nbins+1. Must be monotonically increasing.
     * hdu          Which HDU in the fits file we are interested. Note
     *              that they are indexed from 1, with 1 being the first
     *              "EMISSIVITY" HDU. This is typically HDU 2 in most
     *              FITS files.
     * Z            Atomic number of element of interest (e.g. 6=carbon)
     * rmJ          The ionization stage of interest (1=neutral)
     * doline       If != 0, include line data in spectrum
     * docont       If != 0, include continuum data in spectrum
     *
     * OUTPUTS
     * *spectrum    The emissivity in ph cm3 s-1 bin-1
     *
     */
```

4.2.6 readapec_calc_hdu_ion_spectrum

```
void readapec_calc_hdu_ion_spectrum(struct EMISSION *apec_data,
                            int nbins,
                            double *ebins,
                            int hdu,
                            int Z,
                            int rmJ,
                            double *spectrum);


    /* This routine calculates the spectrum of an ion for a specific HDU.
     * It assumes an abundance of 1.0 for all elements.
     *
     * INPUTS
     * apec_data    all of the emissivity data (returned by
     *              read_apec_outputs)
     * nbins        the number of bins the output spectrum
     * *ebins       the edges of the bins, in keV. Note that the array
     *              length is nbins+1. Must be monotonically increasing.
     * hdu          Which HDU in the fits file we are interested. Note
     *              that they are indexed from 1, with 1 being the first
     *              "EMISSIVITY" HDU. This is typically HDU 2 in most
     *              FITS files.
     * Z            Atomic number of element of interest (e.g. 6=carbon)
     * rmJ          The ionization stage of interest (1=neutral)
     *
     * OUTPUTS
     * *spectrum    The emissivity in ph cm3 s-1 bin-1
     */
```

4.2.7 readapec_find_kT_hdu

```
int readapec_find_kT_hdu(struct EMISSION *apec_data,
                  double kT, int log) {


    /* This routine finds the nearest HDU index for a given temperature
     *
     * INPUTS
     * apec_data    all of the emissivity data (returned by
```

```
     *              read_apec_outputs)
     * kT           target temperature (keV)
     * log          if log=1, then find closest on a log10 scale
     *
     * RETURNS
     *      The hdu nearest to the parameter of interest (in this case kT)
     */
```

4.2.8 readapec_calc_ion_spectrum

```
void readapec_calc_ion_spectrum(struct EMISSION *apec_data,
                        int nbins,
                        double *ebins,
                        double kT,
                        int Z,
                        int rmJ,
                        int doline,
                        int docont
                        int nearest,
                        double *spectrum) {

    /* This routine calculates the spectrum at a specific temperature
     * Setting nearest = 1, it will calculate the spectrum of the nearest
     * HDU, setting it =0, it will interpolate between the 2 adjacent
     * HDUs in log space.
     *
     * INPUTS
     * apec_data    all of the emissivity data (returned by
     *              read_apec_outputs)
     * nbins        the number of bins the output spectrum
     * *ebins       the edges of the bins, in keV. Note that the array
     *              length is nbins+1. Must be monotonically increasing.
     * hdu          Which HDU in the fits file we are interested. Note
     *              that they are indexed from 1, with 1 being the first
     *              "EMISSIVITY" HDU. This is typically HDU 2 in most
     *              FITS files.
     * Z            Atomic number of element of interest (e.g. 6=carbon)
     * rmJ          The ionization stage of interest (1=neutral). If 0,
     *              do all the ions of the element.
     * doline       1:Include line emissions 0: skip
```

```
     * docont       1:Include continuum emissions 0: skip
     * nearest      If 0, interpolate to get exact results.
     *              If 1, just return the sepctrum of the nearest HDU
     * OUTPUTS
     * *spectrum    The emissivity in ph cm3 s-1 bin-1
     * */
```

4.2.9 readapec_calc_allion_spectrum

```
readapec_calc_allion_spectrum(struct EMISSION *apec_data,
                              int nbins,
                              double *ebins,
                              double kT,
                              int nearest,
                              int nelements,
                              int *Zlist,
                              int byion,
                              struct EMISSION_LIST **emission_list);
    /* This routine calculates the spectrum at a specific temperature
     * for all the ions of all the elements in Zlist. It then returns
     * these in an EMISSIONLIST structure, which stores them as separate
     * emissivities. This is useful if you want to later change abundances
     * /
     *
     * INPUTS
     * apec_data    all of the emissivity data (returned by
     *              read_apec_outputs)
     * nbins        the number of bins the output spectrum
     * *ebins       the edges of the bins, in keV. Note that the array
     *              length is nbins+1. Must be monotonically increasing.
     * kT           The temperature requested.
     * nearest      If 0, interpolate to get exact results.
     *              If 1, just return the sepctrum of the nearest HDU
     * nelements    Number of elements to get spectra for
     * Zlist        Atomic numbers of elements of interest (e.g. 6=carbon)
     * byion        If 0, return results for each element summed.
     *              If 1, return results for each ion separately.
     * OUTPUTS
     * emission_list  All of the emissivities in ph cm^3 s-1 bin-1
     * */
```

4.2.10 readapec_calc_total_emission_abundance

```
void readapec_calc_total_emission_abundance(
                                    struct EMISSION_LIST *emission_list,
                                    int nelements,
                                    int *Zlist,
                                    double *abundlist,
                                    double defaultabund,
                                    double *spectrum);

     /* This routine takes a calculated emission list (separated by ion
      * or element) and applies a relative abundance to them. There are
      * already abundances for the solar photosphere from Andres and
      * Grevesse 1989 built into the emissivities, so new abundances
      * should be entered relative to these values */

    /*
     * INPUTS
     * emission_list  All of the emissivities in ph cm^3 s-1 bin-1
     * nelements      The number of elements for which abundances will be
     *                provided. Any elements which are in "emission_list"
     *                but not specified in Zlist will be assumed to have
     *                abundance as specified in unlistedabund
     * Zlist          the elements for which abundances are provided
     * abundlist      the abundances of these elements
     * defaultabund  The default abundances for elements not in abundlist
     * OUTPUTS
     * spectrum       The total spectrum
     * */
```

Index