ChiantiPy: a Python package for Astrophysical Spectroscopy

Will T. Barnes[§*], Kenneth P. Dere[‡]

https://youtu.be/79ledxbnrPU

Abstract—ChiantiPy is an interface to the CHIANTI atomic database for astrophysical spectroscopy. The highly-cited CHIANTI project, now in its 20th year, is an invaluable resource to the solar physics community. The ChiantiPy project brings the power of the scientific Python stack to the CHIANTI database, allowing solar physicists and astronomers to easily make use of this atomic data and calculate commonly used quantities from it such as radiative loss rates and emissivities for particular atomic transitions. This paper will discuss the capabilities of the CHIANTI database and the ChiantiPy project as well as the current state of the project and its place in the solar physics community. We will demonstrate how the core modules in ChiantiPy can be used to study emission from optically thin transitions and the continuum in the x-ray and EUV wavelengths. Additionally, we will discuss some of the infrastructure around the ChiantiPy project and some of the goals for the near future.

Index Terms—solar physics, atomic physics, astrophysics, spectroscopy

Introduction

Nearly all astrophysical observations are done through *remote sensing*. Light at various wavelengths is collected by instruments, either ground- or space-based, in an attempt to understand physical processes happening in distant astrophysical objects. However, in order to translate these detector measurements to meaningful physical insight, it is necessary to understand what physical conditions give rise to different spectral lines and continuum emission. Started in 1996 by researchers at the Naval Research Laboratory, the University of Cambridge, and Arcetri Astrophysical Observatory in Florence for the purpose of analyzing solar spectra, the CHIANTI atomic database provides a set of up-to-date atomic data for thirty different elements as well as a suite of tools, written in the proprietary Interactive Data Language (IDL), for analyzing this data. Described in a series of 15 papers from 1997 to 2016 that have been cited collectively over 3000 times (see Table 1), the CHIANTI database is an invaluable resource to the solar physics community.

The CHIANTI project is comprised of two main parts: the database containing the actual atomic data and the IDL software libraries for accessing the data and calculating useful quantities from them. The database provides atomic data for optically-thin transitions, primarily in the x-ray and extreme ultraviolet (EUV) wavelengths, for ions of 30 different elements, H ($Z = 1$) through Zn ($Z = 30$). The CHIANTI project stemmed largely from the need for a consolidated database of spectral lines for interpreting data from spectroscopic and narrow-band solar observing instruments.

Paper	Version	Citations
[DLM+97]	1	1167
[YLT98]	1	105
[LLD+99]	2	94
[DLYDZ01]	3	156
[LFD02]	3	86
[YDZL+03]	4	250
[LDZY+06]	5	373
[LP06]	5	25
[DLY+09]	6	301
[LY09]	6	25
[YL09]	6	22
[LDZY+12]	7	174
[LYD+13]	7.1	227
[DZDY+15]	8	60
[YDL+16]	8	1
		Total
		3066

TABLE 1: *All publications describing the data and capabilities of the CHIANTI atomic database, the associated version of the database, and the number of citations as reported by the NASA Astrophysics Data System at the time of writing.*

While IDL has been the lingua franca of solar physics for over twenty years, Python is gaining momentum in the community and is the language of choice for many younger researchers. This is largely due to the success of Python in general astronomy (e.g. Astropy), the advent of SunPy, a stable and well-supported Python package for solar data analysis [SMC+15], and the adoption of Python as the language of choice by the Daniel K. Inouye Solar Telescope (DKIST), an instrument expected to be the world's largest solar telescope with an estimated data output of 11 TB per day [WBH+16].

Given the growing popularity of Python in the solar community and the importance of CHIANTI to solar observers and modelers alike, a well-supported Python interface to this database is critical. The ChiantiPy project, started in 2003 by Ken Dere, provides a Python package for interacting with the CHIANTI

* *Corresponding author: will.t.barnes@rice.edu*
§ *Department of Physics and Astronomy, Rice University, Houston, TX, USA*
‡ *Department of Physics and Astronomy, George Mason University, Fairfax, VA, USA*

Filetype	Description
ELVLC	Index and energy for each level
WGFA	Wavelength, radiative decay rates, and oscillator strengths for each transiton
SCUPS	Scaled effective collision strengths for each transition
FBLVL	Energy levels for free-bound continuum calculation

TABLE 2: *Some of the filetypes available for each ion in the CHIANTI database. Adapted from Table 1 of [YDL+16].*

database and an alternative to the IDL tools. ChiantiPy is not a direct translation of its IDL counterpart, but instead provides an intuitive object oriented interface to the database (compared to the more functional approach in IDL). ChiantiPy provides an easy to use API to the raw atomic data in the CHIANTI database as well as Python versions of all the primary calculations performed by the original IDL software, including the level balance equation and the ionization equilibrium calculation. This paper will give a brief overview of the CHIANTI database and demonstrate the core capabilities of the ChiantiPy package. These include the line emission for transitions and continuum emission of particular ions as well as spectra and radiative loss rates summed over many ions. We will also discuss the infrastructure of the package and plans for the future of the package.

Database

The CHIANTI database is organized as a collection of directories and ASCII files that can be downloaded as a tarball from the CHIANTI database website or as part of the SolarSoftware (or SolarSoft) IDL package [FH98]. The solar physics community has typically relied on the latter as SolarSoft has served as the main hub for solar data analysis software for the last several decades. SolarSoft provides routines for updating software packages automatically and so traditionally CHIANTI users have updated their distributions, including both the software and the database, in this manner[1].

The structure of the CHIANTI database is such that each top level directory represents an element and each subdirectory is an ion of that element. Files in each of the subdirectories contain pieces of information attached to each ion. The database generally follows the structure `{el}/{el}_{ion}/{el}_{ion}.{filetype}`, where `el` is the element, `ion` is the ion number, and `filetype` is the file extension. For example, the energy level information for Fe V is in the file `fe/fe_5/fe_5.elvlc`. A few of these filetypes are summarized in Table 2. For a complete description of all of the different filetypes available, see Table 1 of [YDL+16] and the CHIANTI user guide. Fig. 1 shows all of the available ions in the CHIANTI database as well as the number of levels available for each ion.

ChiantiPy provides several low-level functions for reading raw data directly from the CHIANTI database. For example, to find the energy of the emitted photon for each transition for Fe V (i.e. the fifth ionization state of iron), you would first read in level information for each transition for a given ion,

```
import ChiantiPy.tools.util as ch_util
fe5_wgfa = ch_util.wgfaRead('fe_5')
ilvl1 = np.array(fe5_wgfa['lvl1']) - 1
ilvl2 = np.array(fe5_wgfa['lvl2']) - 1
```

and then use the indices of the level to find the associated level energies in the ELVLC data,

```
fe5_elvlc = ch_util.elvlcRead('fe_5')
delta_energy = (np.array(fe5_elvlc['ecm'])[ilvl2]
                - np.array(fe5_elvlc['ecm'])[ilvl1])
```

where the associated energy levels are given in cm^{-1}. In general, these functions are only used internally by the core ChiantiPy objects. However, users who need access to the raw data may find them useful.

In addition to each of the files associated with each ion, CHIANTI also provides abundance and ionization equilibrium data for each *element* in the database. The elemental abundance, $N(X)/N(H)$ (i.e. the number of atoms of element X relative to the number of hydrogen atoms), in the corona and photosphere has been measured by many workers and these various measurements have been collected in the CHIANTI atomic database. For example, to read the abundance of Fe as measured by [FMS+92],

```
import ChiantiPy.tools.io as ch_io
ab = ch_io.abundanceRead('sun_coronal_1992_feldman')
fe_ab = abundance['abundance'][ch_util.el2z('Fe')-1]
```

As with the other CHIANTI data files, the abundance values are typically read internally and then exposed to the user through more abstract objects like the `ion` class so reading them in this way is not necessary. Similarly, the ionization equilibrium of each ion of each element is available as a function of temperature and various sets of ionization equilibria data can be used. More details about the ionization equilibrium can be found in later sections.

Default values for the abundance and ionization equilibrium files as well as the units for wavelength (nm, Å, or eV) and energy (ergs or photons) can be set in the users `chiantirc` file, located in `~/.chianti/chiantirc`. These settings are stored in `ChiantiPy.tools.data.Defaults` and can be changed at anytime.

Unless otherwise noted, all quantities are expressed in the cgs unit system, with the exception of wavelengths which are recorded in angstroms (Å). As discussed above, some energies in the CHIANTI atomic database, particularly those pertaining to levels in an atom, are stored in cm^{-1} for convenience (i.e. with $h = c = 1$, a common convention in atomic physics). Results of any calculation in ChiantiPy will always be returned in cgs units (unless explicitly stated in the `chiantirc` file, e.g. photons instead of ergs).

Common Calculations and API

The majority of the ChiantiPy codebase is divided into two modules: `tools` and `core`. The former contains utility and helper functions that are mostly for internal use. The latter contains the primary objects for interacting with the data in the CHIANTI atomic database and performing many common calculations with these data. A summary of the objects in `core` can be found in Table 3. These objects can be roughly divided into two categories: those that deal with information and calculations about individual ions and those that aggregate information over a range of ions in order to perform some calculation. The `ion` and `Continuum`

1. The easiest way to acquire the CHIANTI database is to download and unpack the tarball at http://www.chiantidatabase.org/chianti_download.html. In order for ChiantiPy to find the database, it is necessary to point the `XUVTOP` environment variable to the top of the CHIANTI directory tree. For example, if the database is downloaded to `$HOME/chianti`, `export XUVTOP=$HOME/chianti/dbase` should be placed in the Bash shell configuration file.

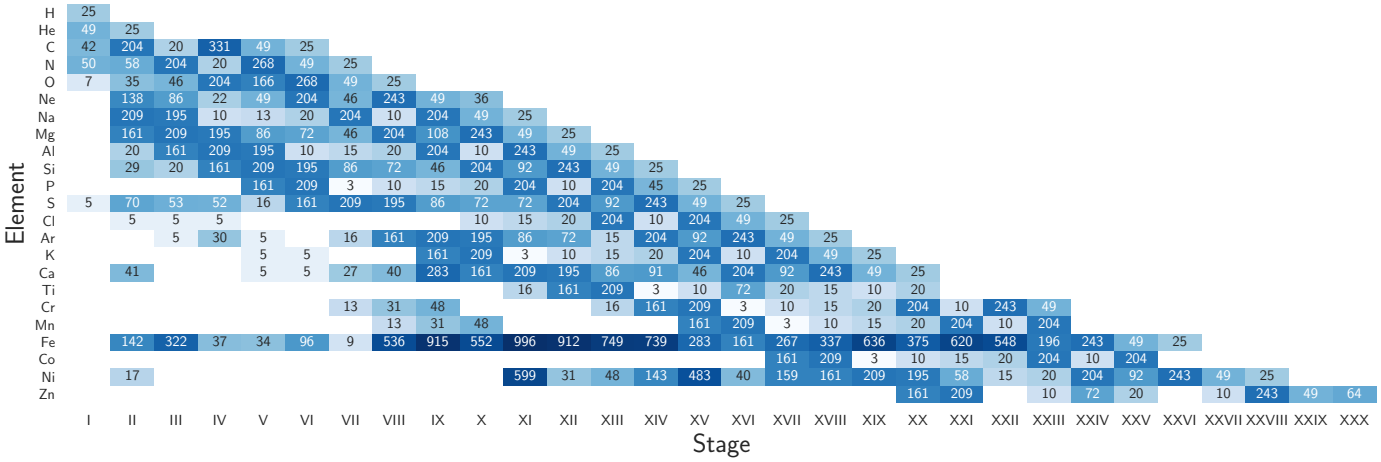

Fig. 1: *All ions available in the latest version (v8.0.6) of the CHIANTI atomic database. The color and number in each square indicate the number of available levels in the database. Adapted from Fig. 1 of [YDL+16].*

Object Name	Description
`ion`	Holds ion properties and calculates level populations and emissivity
`Continuum`	Free-free and free-bound continuum for individual ions
`ioneq`	Ionization equilibrium for individual elements
`spectrum`	Calculate synthetic spectra for a range of ions
`radLoss`	Total radiative losses from multiple ions, including continuum

TABLE 3: *The primary objects in the public API of ChiantiPy.*

objects calculate emissivity information related to specific ions while `ioneq`, `spectrum`, and `radLoss` require information from multiple ions and/or elements.

Line Emission

The most essential and actively developed portion of the ChiantiPy package is the `ion` object which provides an interface to the data and associated calculations for each ion in the database. The `ion` object is initialized with an ion name, a temperature range, and a density[2],

```
import ChiantiPy.core as ch
import numpy as np
temperature = np.logspace(4,6,100)
density = 1e9
fe_5 = ch.ion('fe_5',temperature,density)
```

In this example, we've initialized an `ion` object for Fe V over a temperature range of $T = 10^4 - 10^6$ K at a constant electron density of $n_e = 10^9$ cm^{-3}. All of the data discussed in the previous section are available as attributes of the `ion` object (e.g. `.Elvlc` and `.Wgfa` are dictionaries holding the various fields available in the corresponding filetypes listed in Table 3). In general, ChiantiPy objects follow the convention that methods are lowercase and return their value(s) to attributes with corresponding uppercase names[3]. For example, the abundance value of Fe is stored in `fe_5.Abundance` and the ionization equilibrium is calculated using the method `fe_5.ioneqOne()` with the value being returned to the attribute `fe_5.IoneqOne`.

One of the most often used calculations in CHIANTI and ChiantiPy is the energy level populations as a function of temperature. When calculating the energy level populations in a low density, high temperature, optically-thin plasma, collisional excitation and subsequent decay often occur much more quickly than ionization and recombination, allowing these two processes to be decoupled. Furthermore, it is assumed that all transitions occur between the excited state and the ground state. These two assumptions make up what is commonly known as the *coronal model approximation*. Thus, the level balance equation can be written as,

$$\sum_{k>j} N_k A_{kj} + n_e \sum_{i=j} N_j C_{ij} - \left(\sum_{i<j} N_j A ji + n_e \sum_{k=j} N_j C_{jk} \right) = 0,$$

where A_{kj} is the radiative decay rate, C_{jk} is the collisional excitation coefficient, and N_j is the number of electrons in excited state j [YDL+16]. Since A and C are given by the CHIANTI database, this expression can be solved iteratively to find $n_j = N_j / \sum_j N_j$, the fraction of electrons in excited state j or the level population fraction. Proton excitation rates, primarily between fine structure levels, can also be included in the calculation of n_j. See Eq. 4 of [YDZL+03].

The method `fe_5.populate()` can then be used to calculate the level populations for Fe V. This method populates the `fe_5.Population` attribute and a 100×34 array (i.e. number of temperatures by number of energy levels) is stored in `fe_5.Population['population']`. ChiantiPy also provides the convenience method `fe_5.popPlot()` which provides a quick visualization of level population as a function of temperature for several of the most populated levels. Note that this calculation can be quite expensive for large temperature/density arrays and for ions with many transitions. The left panel of Fig. 2 shows the level population as a function of temperature, $n_j(T)$, for all of the energy levels of Fe V in the CHIANTI database.

2. A single temperature and an array of densities is also valid. The only requirement is that if one or the other is not of length 1, both arrays must have the same length. The ion object can also be initialized without any temperature or density information if only the ion data is needed.

3. This convention is likely to change in the near future as the ChiantiPy codebase is brought into compliance with the PEP 8 Style Guide for Python code.

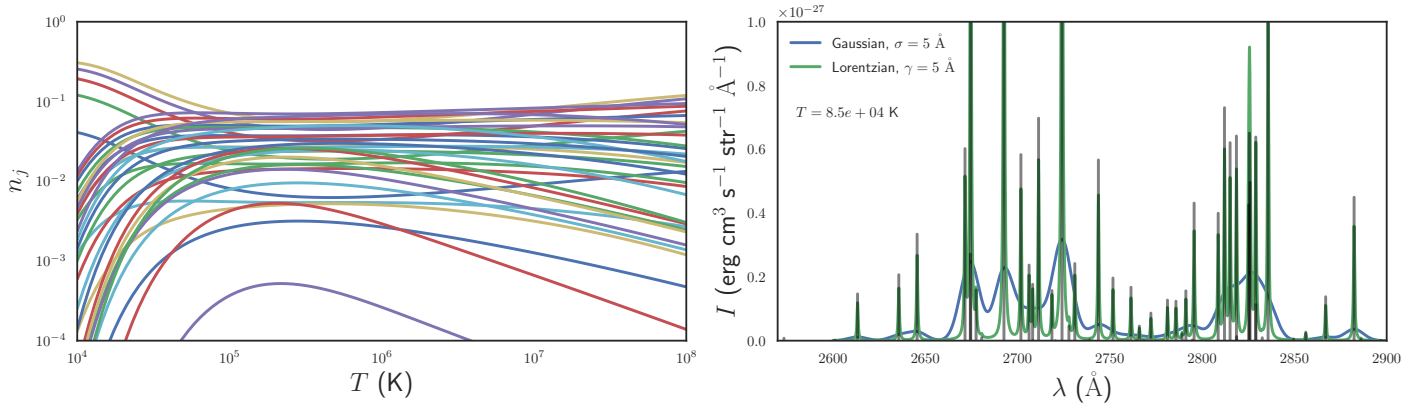

Fig. 2: *Level populations, n_j, as a function of temperature (left) and intensity as a function of wavelength (right) for Fe V. The various curves in the left panel represent the multiple energy levels of the Fe V ion. The right panel shows the intensity at the discrete wavelength values (black) as well as the spectra folded through a Gaussian filter with $\sigma = 5$ Å (blue) and a Lorentzian filter with $\gamma = 5$ Å (green).*

When dealing with spectral line emission, we are often most interested in the line *intensity*, that is, the power per unit volume as a function of temperature (and density). For a particular transition λ_{ij}, the line intensity can be written as,

$$I_{ij} = \frac{1}{4\pi}\frac{hc}{\lambda}\mathrm{Ab}(X)X_k A_{ij} n_j n_e^{-1}, \quad [\mathrm{erg\,cm^3\,s^{-1}\,sr^{-1}}]$$

where $\mathrm{Ab}(X)$ is the abundance and X_k is the ionization equilibrium. To calculate the intensity for each transition in CHIANTI for Fe V, we can use the method `fe_5.intensity()` which returns a 100×219 array (i.e. dimension of temperature by the number of available transitions). The convenience methods `fe_5.intensityPlot()` and `fe_5.intensityList()` can also be used to quickly visualize and enumerate the most intense lines produced by the ion, respectively.

Finally, the intensity can be convolved with a filter to calculate the intensity as a *continuous* function of wavelength to simulate an observed *spectrum*. For a single ion this is done using the `fe_5.spectrum()` method (see later sections for creating multi-ion spectra). To create a spectrum for Fe V between 2600 Å and 2900 Å,

```
wavelength = np.arange(2.6e3,2.9e3,0.1)
fe_5.spectrum(wavelength)
```

This method also accepts an optional keyword argument for specifying a filter with which to convolve the intensity. The default filter is a Gaussian though `ChiantiPy.tools.filters` includes several different filters including Lorentzian and Boxcar filters. The right panel of Fig. 2 shows the Fe V intensity (black) and spectrum folded through a Gaussian (blue) and Lorentzian (green) filter at the temperature at which the ionization fraction is maximized, $T \approx 8.5 \times 10^4$ K. Similar to the `fe_5.populate()` and `fe_5.intensity()`, ChiantiPy also provides the convenience method `fe_5.spectrumPlot()` for quickly visualizing a spectrum.

Continuum Emission

In addition to calculating emissivities for individual spectral lines, ChiantiPy also calculates the free-free, free-bound, and two-photon continua as a function of wavelength and temperature for each ion. In particular, the `Continuum` object is used to calculate the free-free and free-bound emissivities. Free-free emission (or *bremsstrahlung*) is produced by collisions between free electrons and positively charged ions. The free-free emissivity is given by,

$$\frac{dW}{dtdVd\lambda} = \frac{c}{3m_e}\left(\frac{\alpha h}{\pi}\right)^3\left(\frac{2\pi}{3m_e k_B}\right)^{1/2}\frac{Z^2}{\lambda^2 T^{1/2}}\bar{g}_{ff} \times \exp\left(-\frac{hc}{\lambda k_B T}\right), \quad [\mathrm{erg\,cm^3\,s^{-1}\,Å^{-1}\,sr^{-1}}]$$

where α is the fine structure constant, Z is the nuclear charge, T is the electron temperature, and $\bar{g}_{ff}$ is the velocity-averaged Gaunt factor [RL79]. $\bar{g}_{ff}$ is calculated using the methods of [ISK+00] (`Continuum.itoh_gaunt_factor()`) and [Sut98] (`Continuum.sutherland_gaunt_factor()`), depending on the temperature range.

Similarly, free-bound emission is produced when a free electron collides with a positively-charged ion and the previously-free electron is captured into an excited state of the ion. Because this process (unlike free-free emission) involves the details of the energy level structure of the ion, its formulation is necessarily quantum mechanical though a semi-classical treatment is possible (see Section 4.7.2 of [PFL08] and Section 10.5 of [RL79]). From [YDZL+03], the free-bound emission can be calculated as,

$$\frac{dW}{dtdVd\lambda} = \frac{1}{4\pi}\frac{2}{hk_B c^3 m_e\sqrt{2\pi k_B m_e}}\frac{E^5}{T^{3/2}}\sum_i \frac{\omega_i}{\omega_0}\sigma_i^{bf} \times \exp\left(-\frac{E-I_i}{k_B T}\right), \quad [\mathrm{erg\,cm^3\,s^{-1}\,Å^{-1}\,sr^{-1}}]$$

where $E = hc/\lambda$ is the photon energy, ω_i and ω_0 are the statistical weights of the i^{th} level of the recombined ion and the ground level of the recombining ion, respectively, σ_i^{bf} is the photoionization cross-section, and I_i is the ionization potential of level i. The cross-sections are calculated using the methods of [VY95] (for the ground state, i.e. $i = 0$) and [KL61] (for $i \neq 0$). An optional `use_verner` keyword argument (`True` by default) is included in the `Continuum.calclulate_free_bound_emission()` so that users can choose to only use the method of [KL61] in the photoionization cross-section calculation.

To calculate the free-free and free-bound emission with ChiantiPy,

```
temperature = np.logspace(6,8.5,100)
cont_fe18 = ch.Continuum('fe_18',temperature)
wavelength = np.logspace(0,3,100)
```

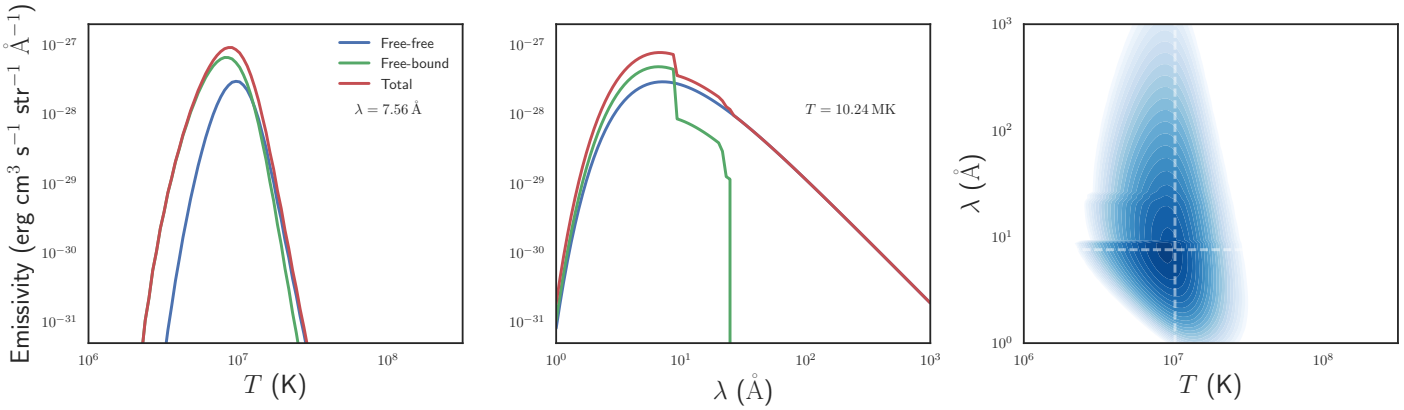

Fig. 3: *Continuum emission for Fe XVIII. The left (middle) panel shows the free-free, free-bound, and total emission as a function of temperature (wavelength) for $\lambda \approx 7.5$ Å ($T \approx 10^7$ K). The contours in the rightmost panel show the total emissivity as a function of both temperature and wavelength on a log scale. The dashed lines indicate the cuts shown in the left and middle panels.*

```
cont_fe18.calculate_free_free_emission(wavelength)
cont_fe18.calculate_free_bound_emission(wavelength)
```

The `Continuum.calculate_free_free_emission()` (`Continuum.calculate_free_bound_emission()`) method stores the N_T by N_λ array (e.g. in the above example, 100×100) in the `Continuum.free_free_emission` (`Continuum.free_bound_emission`) attribute. The left and middle panels of Fig. 3 show the free-free, free-bound, and total continuum emission as a function of temperature and wavelength, respectively, and the rightmost panel shows the total continuum emission as a function of both temperature and wavelength for the Fe XVIII ion.

The `Continuum` object also provides methods for calculating the free-free and free-bound radiative losses (i.e. the wavelength-integrated emission). These methods are primarily used by the `radiativeLoss` module. The `Continuum` module has recently been completely refactored and validated against the corresponding IDL results.

A contribution from the two-photon continuum can also be calculated with ChiantiPy though this is included in the `ion` object through the method `ion.twoPhoton()`. The two-photon continuum calculation is included in the `ion` object and not the `Continuum` object because the level populations are required when calculating the two-photon emissivity. See Eq. 11 of [YDZL+03].

Ionization Equilibrium

The ionization equilibrium of a particular ion describes what fraction of the ions of an element are in a particular ionization state at a given temperature. Specifically, the ionization equilibrium is determined by the balance of ionization and recombination rates. For an element X and an ionization state i, assuming ionization equilibrium, the ionization state $X_i = N(X^{+i})/N(X)$ is given by,

$$I_{i-1}X_{i-1} + R_i X_{i+1} = I_i X_i + R_{i-1} X_i$$

where I_i and R_i are the total ionization and recombination rates for ionization state i, respectively. In CHIANTI, these rates are assumed to be density-independent and only a function of temperature.

In ChiantiPy, the ionization equilibrium for a particular element can be calculated using the `ioneq` module,

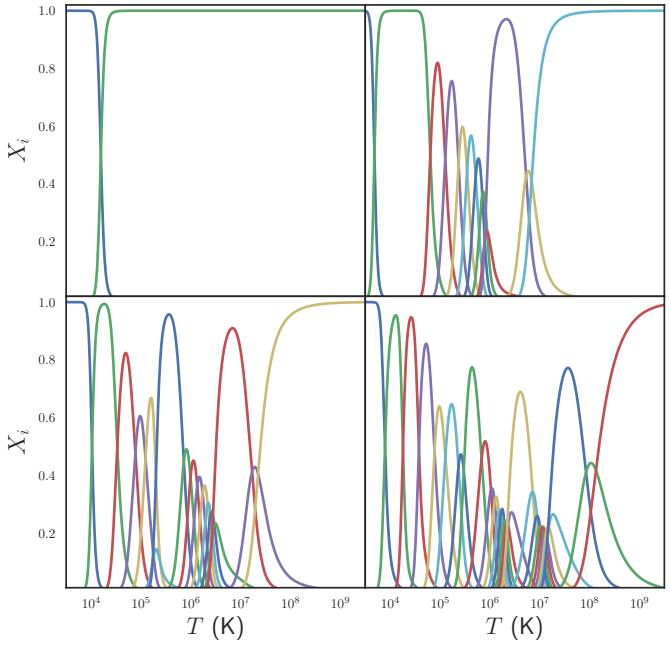

Fig. 4: *Population fractions as a function of temperature for (clockwise from upper left) H, Na, Fe, and S calculated using ionization and recombination data from CHIANTI and assuming ionization equilibrium.*

```
fe_ioneq = ch.ioneq('Fe')
temperature = np.logspace(3.5,9.5,500)
fe_ioneq.calculate(temperature)
```

The `ioneq.calculate()` method sets the `Ioneq` attribute, an array with $Z+1$ columns and N_T rows, where N_T is the length of the temperature array. In the example above, `fe_ioneq.Ioneq` has 27 rows (i.e. $Z = 26$ for Fe) and 500 columns. Fig. 4 shows the ion population fractions for four different elements as a function of temperature, assuming ionization equilibrium.

The `ioneq` module also allows the user to load a predefined set of ionization equilibria via the `ioneq.load()` method. Though CHIANTI includes several ionization equilibrium datasets from other workers, it is recommended to use the most up to date version supplied by CHIANTI (see [DLY+09] for more details).

To load the ionization equilibrium data for Fe,

```
fe_ioneq = ch.ioneq('Fe')
fe_ioneq.load()
```

This will populate the `fe_ioneq.Temperature` and `fe_ioneq.Ioneq` attributes with data from the appropriate ionization equilibrium file. By default, this will be `ioneq/chianti.ioneq` unless otherwise specified in the `chiantirc` file or the `ioneqName` keyword argument.

Spectra

In addition to being able to calculate spectra for single ions, ChiantiPy also provides a wrapper for calculating composite spectra for a range of ions, including continuum contributions. This is handled through the `spectrum` object. To calculate a composite spectrum in ChiantiPy,

```
temperature = np.array([1e+6,4e+6,1e+7])
density = 1e9
wavelength = np.linspace(10,100,1000)
min_abund = 1e-4
spec = ch.spectrum(temperature, density,
                   wavelength, minAbund=min_abund)
```

The spectrum as a continuous function of wavelength can then be accessed in the `spec.Spectrum['intensity']` attribute as a $N_T \times N_\lambda$ array (i.e. 3×1000 in the above example. Most of the keywords that can be passed to `ion.spectrum()` can also be passed to `ChiantiPy.spectrum()` and the attributes that are available following the calculation are largely the same. Fig. 5 shows the integrated spectrum as calculated above with several of the included transitions labeled.

Because of the need to perform calculations and aggregate data over a large range of ions, running `ChiantiPy.spectrum()` can be very time consuming, particularly for large temperature/density ranges. The above code snippet takes approximately five minutes to execute on a modern desktop machine. To help mitigate this difficulty, ChiantiPy provides a parallelized version of the `ChiantiPy.spectrum` module called `ChiantiPy.mspectrum`[4] which takes advantage of the `multiprocessing` package and can help to speed up the calculation, particularly on machines with many cores. The interface to the parallelized code is largely the same as the serial version.

Radiative Losses

The radiative loss rate is an important quantity for calculating the energy loss in coronal plasmas, particularly in hydrodynamic simulations of coronal loops. The total radiative loss rate is given by,

$$\Lambda = \Lambda_{continuum} + \Lambda_{line}, \quad [\mathrm{erg\,cm^3\,s^{-1}}]$$

where

$$\Lambda_{line} = \sum_X \Lambda_X = \sum_{X,k} \Lambda_{X_k} = \sum_{X,k,\lambda_{ij}} \Lambda_{X_{k,\lambda_{ij}}}$$
$$= \sum_{X,k,\lambda_{ij}} \mathrm{Ab}(X) X_k \frac{hc}{\lambda} A_{ij} n_j n_e^{-1},$$

is the contribution to the radiative losses summed over every element (X), ion (X_k) and transition (λ_{ij}), and $\Lambda_{continuum}$ includes the free-free, free-bound, and two-photon continuum contributions to the radiative loss.

4. ChiantiPy provides an additional module `ChiantiPy.ipymspectrum` to support parallelized spectrum calculations inside the Jupyter notebook and qtconsole.

In ChiantiPy, the radiative loss rate can be calculated using the `radLoss` module for a particular temperature and density range. To calculate the total radiative loss rate for all ions with an abundance greater than 10^{-4},

```
temperature = np.logspace(4,8,100)
rl = ch.radLoss(temperature, 1e9, minAbund=1e-4)
```

Instantiating the `radLoss` object automatically calculates the radiative loss rate and stores the total loss rate in `rl.RadLoss['rate']`, in this case an array of length 100. If the continuum contributions are included (`doContinuum` is `True` by default), the free-free, free-bound, and two-photon components are stored in `rl.FreeFreeLoss`, `rl.FreeBoundLoss`, and `rl.TwoPhotonLoss`, respectively. Ions with low abundances can be excluded with the `minAbund` keyword argument which can speed up the calculation. A custom abundance dataset can also be set with the `abundance` keyword. Note that the above calculation takes approximately 11 minutes on modern hardware. Fig. 6 shows the total radiative losses using the coronal abundances of [FMS+92] (solid) and the photospheric abundances of [AGSS09] (dashed). The coronal abundance case is also broken down into the line emission, free-free, free-bound, and two-photon continuum components.

Documentation, Testing, and Infrastructure

The ChiantiPy project has made an effort to embrace modern development practices when it comes to developing, documenting and releasing the ChiantiPy codebase. Like many open source projects started in the late 2000s, ChiantiPy was originally hosted on SourceForge, but has now moved its development entirely to GitHub. The SVN commit history is in the process of being migrated to GitHub as well. The move to GitHub has provided increased development transparency, ease of contribution, and better integration with third-party services.

An integral part of producing quality scientific code, particularly that meant for a large user base, is continually testing said code as improvements are made and features are added. For each merge into master as well as each pull request, a series of tests is run on Travis CI, a continuous integration service that provides free and automated builds configured through GitHub webhooks. This allows each contribution to the codebase to be tested to ensure that these changes do not break the codebase in unexpected ways. Currently, ChiantiPy is tested on Python 2.7, 3.4, and 3.5, with full 3.6 support expected soon. Currently, the ChiantiPy package is installed in each of these environments and minimal set of tests of each core module is run. The actual module tests are currently quite sparse though one of the more pressing goals of the project is to increase test coverage of the core modules.

One of the most important parts of any codebase is the documentation. The ChiantiPy documentation is built using Sphinx and is hosted on Read the Docs. At each merge into the master branch, a new Read the Docs build is kicked off, ensuring that the ChiantiPy API documentation is never out of date with the most recent check in. In addition to the standard API documentation, the ChiantiPy Read the Docs page also provides a tutorial for using the various modules in ChiantiPy as well as a guide for those switching from the IDL version.

ChiantiPy has benefited greatly from the astropy-helpers package template provided by the Astropy collaboration [ART+13].

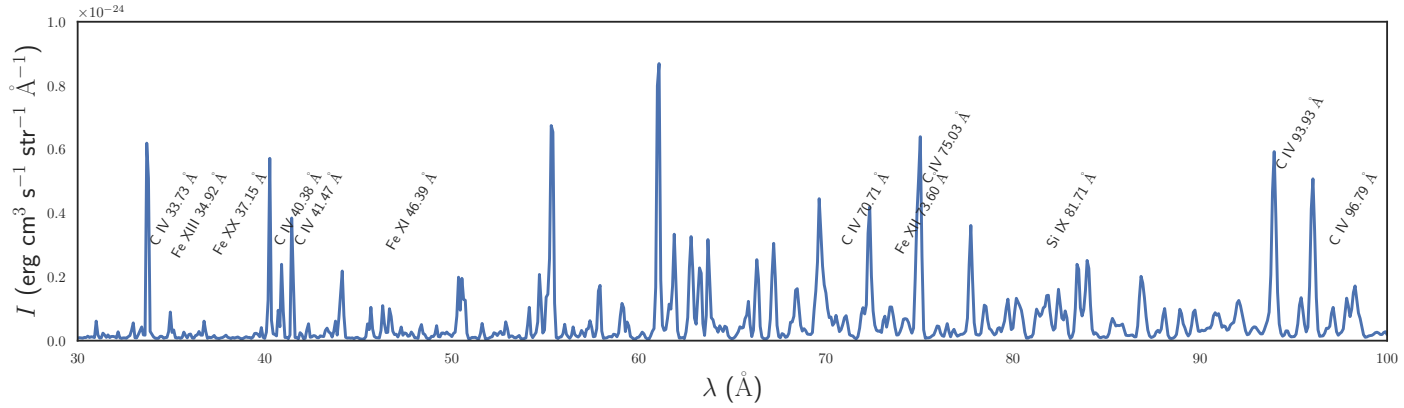

Fig. 5: *Total spectrum for all ions with an abundance greater than* 10^{-4}*, including the continuum, integrated over three temperatures,* $T = 10^6, 4\times10^6, 10^7$ *K and at a constant density of* $n = 10^9$ cm^{-3}*. A few of the transitions included in the spectrum are denoted by their respective spectroscopic labels and wavelengths.*

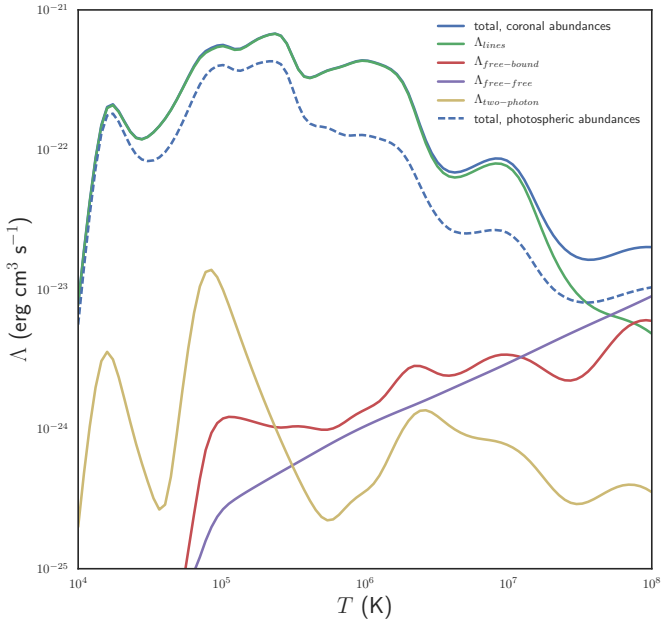

Fig. 6: *Combined radiative losses for all ions in the CHIANTI database for coronal abundances (blue,solid) and photospheric abundances (blue, dashed). The coronal abundance case is also broken down into the line emission (green) and free-free (purple), free-bound (red), and two-photon (yellow) continuum components. In the coronal case, the minimum abundance for elements to be included in the calculation is* 10^{-4} *and* 10^{-6} *for the photospheric case.*

asropy-helpers provides boilerplate code for setting up documentation and testing frameworks which has allowed the package to adopt modern testing and documentation practices with little effort.

Conclusion

In this paper, we have described the main capabilities of ChiantiPy, a package for astrophysical spectroscopy and an interface to the widely used and highly cited CHIANTI atomic database. ChiantiPy provides basic functions for reading the raw data as well as higher-level abstractions (e.g. the `ion` class) for exploring the data and performing common calculations with them. ChiantiPy also provides modules for calculating continuum emission, synthesizing spectra, and calculating radiative loss curves. The project has recently made significant infrastructure improvements by moving development to GitHub, adding automatic documentation builds, and implementing a minimal test suite. Future improvements include the addition of unitful quantities throughout the codebase (e.g. the Astropy unit system) and increased test coverage.

References

[AGSS09] Martin Asplund, Nicolas Grevesse, A. Jacques Sauval, and Pat Scott. The Chemical Composition of the Sun. *Annual Review of Astronomy and Astrophysics*, 47:481–522, September 2009. doi:10.1146/annurev.astro.46.060407.145222.

[ART+13] Astropy Collaboration, Thomas P. Robitaille, Erik J. Tollerud, Perry Greenfield, Michael Droettboom, Erik Bray, Tom Aldcroft, Matt Davis, Adam Ginsburg, Adrian M. Price-Whelan, Wolfgang E. Kerzendorf, Alexander Conley, Neil Crighton, Kyle Barbary, Demitri Muna, Henry Ferguson, Frédéric Grollier, Madhura M. Parikh, Prasanth H. Nair, Hans M. Unther, Christoph Deil, Julien Woillez, Simon Conseil, Roban Kramer, James E. H. Turner, Leo Singer, Ryan Fox, Benjamin A. Weaver, Victor Zabalza, Zachary I. Edwards, K. Azalee Bostroem, D. J. Burke, Andrew R. Casey, Steven M. Crawford, Nadia Dencheva, Justin Ely, Tim Jenness, Kathleen Labrie, Pey Lian Lim, Francesco Pierfederici, Andrew Pontzen, Andy Ptak, Brian Refsdal, Mathieu Servillat, and Ole Streicher. Astropy: A community Python package for astronomy. *Astronomy and Astrophysics*, 558:A33, October 2013. doi:10.1051/0004-6361/201322068.

[DLM+97] K. P. Dere, E. Landi, H. E. Mason, B. C. Monsignori Fossi, and P. R. Young. CHIANTI - an atomic database for emission lines - I. Wavelengths greater than 50 Å. *Astronomy and Astrophysics Supplement Series*, 125(1):25, 1997. doi:10.1051/aas:1997368.

[DLY+09] K. P. Dere, E. Landi, P. R. Young, G. Del Zanna, M. Landini, and H. E. Mason. CHIANTI - an atomic database for emission lines. IX. Ionization rates, recombination rates, ionization equilibria for the elements hydrogen through zinc and updated atomic data. *Astronomy and Astrophysics*, 498:915–929, May 2009. doi:10.1051/0004-6361/200911712.

[DLYDZ01] K. P. Dere, E. Landi, P. R. Young, and G. Del Zanna. CHIANTI-An Atomic Database for Emission Lines. IV. Extension to X-Ray Wavelengths. *The Astrophysical Journal Supplement Series*, 134:331–354, June 2001. doi:10.1086/320854.

[DZDY+15] G. Del Zanna, K. P. Dere, P. R. Young, E. Landi, and H. E. Mason. CHIANTI - An atomic database for emission lines. Version 8. *Astronomy and Astrophysics*, 582:A56, October 2015. doi:10.1051/0004-6361/201526827.

[FH98] S. L. Freeland and B. N. Handy. Data Analysis with the SolarSoft System. *Solar Physics*, 182:497–500, October 1998. doi:10.1023/A:1005038224881.

[FMS+92] U. Feldman, P. Mandelbaum, J. F. Seely, G. A. Doschek, and H. Gursky. The potential for plasma diagnostics from stellar extreme-ultraviolet observations. *The Astrophysical Journal Supplement Series*, 81:387–408, July 1992. doi:10.1086/191698.

[ISK+00] Naoki Itoh, Tsuyoshi Sakamoto, Shugo Kusano, Satoshi Nozawa, and Yasuharu Kohyama. Relativistic Thermal Bremsstrahlung Gaunt Factor for the Intracluster Plasma. II. Analytic Fitting Formulae. *The Astrophysical Journal Supplement Series*, 128:125–138, May 2000. doi:10.1086/313375.

[KL61] W. J. Karzas and R. Latter. Electron Radiative Transitions in a Coulomb Field. *The Astrophysical Journal Supplement Series*, 6:167, May 1961. doi:10.1086/190063.

[LDZY+06] E. Landi, G. Del Zanna, P. R. Young, K. P. Dere, H. E. Mason, and M. Landini. CHIANTI-An Atomic Database for Emission Lines. VII. New Data for X-Rays and Other Improvements. *The Astrophysical Journal Supplement Series*, 162:261–280, January 2006. doi:10.1086/498148.

[LDZY+12] E. Landi, G. Del Zanna, P. R. Young, K. P. Dere, and H. E. Mason. CHIANTI—An Atomic Database for Emission Lines. XII. Version 7 of the Database. *The Astrophysical Journal*, 744:99, January 2012. doi:10.1088/0004-637X/744/2/99.

[LFD02] E. Landi, U. Feldman, and K. P. Dere. CHIANTI-An Atomic Database for Emission Lines. V. Comparison with an Isothermal Spectrum Observed with SUMER. *The Astrophysical Journal Supplement Series*, 139:281–296, March 2002. doi:10.1086/337949.

[LLD+99] E. Landi, M. Landini, K. P. Dere, P. R. Young, and H. E. Mason. CHIANTI - an atomic database for emission lines. III. Continuum radiation and extension of the ion database. *Astronomy and Astrophysics Supplement Series*, 135:339–346, March 1999. doi:10.1051/aas:1999449.

[LP06] E. Landi and K. J. H. Phillips. CHIANTI-An Atomic Database for Emission Lines. VIII. Comparison with Solar Flare Spectra from the Solar Maximum Mission Flat Crystal Spectrometer. *The Astrophysical Journal Supplement Series*, 166:421–440, September 2006. doi:10.1086/506180.

[LY09] E. Landi and P. R. Young. CHIANTI—An Atomic Database for Emission Lines. X. Spectral Atlas of a Cold Feature Observed with Hinode/EUV Imaging Spectrometer. *The Astrophysical Journal*, 706:1–20, November 2009. doi:10.1088/0004-637X/706/1/1.

[LYD+13] E. Landi, P. R. Young, K. P. Dere, G. Del Zanna, and H. E. Mason. CHIANTI—An Atomic Database for Emission Lines. XIII. Soft X-Ray Improvements and Other Changes. *The Astrophysical Journal*, 763:86, February 2013. doi:10.1088/0004-637X/763/2/86.

[PFL08] Kenneth J. H. Phillips, Uri Feldman, and Enrico Landi. *Ultraviolet and X-Ray Spectroscopy of the Solar Atmosphere*. June 2008.

[RL79] George B. Rybicki and Alan P. Lightman. *Radiative Processes in Astrophysics*. New York : Wiley, 1979.

[SMC+15] SunPy Community, Stuart J. Mumford, Steven Christe, David Pérez-Suárez, Jack Ireland, Albert Y. Shih, Andrew R. Inglis, Simon Liedtke, Russell J. Hewett, Florian Mayer, Keith Hughitt, Nabil Freij, Tomas Meszaros, Samuel M. Bennett, Michael Malocha, John Evans, Ankit Agrawal, Andrew J. Leonard, Thomas P. Robitaille, Benjamin Mampaey, Jose Iván Campos-Rozo, and Michael S. Kirk. SunPy—Python for solar physics. *Computational Science and Discovery*, 8:014009, January 2015. doi:10.1088/1749-4699/8/1/014009.

[Sut98] Ralph S. Sutherland. Accurate free-free Gaunt factors for astrophysical plasmas. *Monthly Notices of the Royal Astronomical Society*, 300:321–330, October 1998. doi:10.1046/j.1365-8711.1998.01687.x.

[VY95] D. A. Verner and D. G. Yakovlev. Analytic FITS for partial photoionization cross sections. *Astronomy and Astrophysics Supplement Series*, 109, January 1995.

[WBH+16] Fraser T. Watson, Steven J. Berukoff, Tony Hays, Kevin Reardon, Daniel J. Speiss, and Scott Wiant. Calibration development strategies for the Daniel K. Inouye Solar Telescope (DKIST) data center. volume 9910, pages 99101G–99101G–11, 2016. doi:10.1117/12.2233179.

[YDL+16] P. R. Young, K. P. Dere, E. Landi, G. Del Zanna, and H. E. Mason. The CHIANTI atomic database. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 49(7):074009, 2016. doi:10.1088/0953-4075/49/7/074009.

[YDZL+03] P. R. Young, G. Del Zanna, E. Landi, K. P. Dere, H. E. Mason, and M. Landini. CHIANTI-An Atomic Database for Emission Lines. VI. Proton Rates and Other Improvements. *The Astrophysical Journal Supplement Series*, 144:135–152, January 2003. doi:10.1086/344365.

[YL09] P. R. Young and E. Landi. Chianti—An Atomic Database for Emission Lines. XI. Extreme-Ultraviolet Emission Lines of Fe VII, Fe VIII, and Fe IX Observed by Hinode/EIS. *The Astrophysical Journal*, 707:173–192, December 2009. doi:10.1088/0004-637X/707/1/173.

[YLT98] P. R. Young, E. Landi, and R. J. Thomas. CHIANTI: An atomic database for emission lines. II. Comparison with the SERTS-89 active region spectrum. *Astronomy and Astrophysics*, 329:291–314, January 1998.