Synple User's Guide

C. Allende Prieto, I. Hubeny, T. Lanz, Y. M. Osorio

December 21, 2023

`Synple` is a Python interface to the spectral synthesis code Synspec and a set of related utilities. Synple can compute emerging spectra from LTE Kurucz, MARCS, Phoenix or Tlusty model atmospheres, as well as NLTE Tlusty models. Appropriate default choices for the continuum and line opacities are provided, while the chemical abundances and micro-turbulence velocity are taken from the input model atmospheres, but can be changed as needed.

Contents

1 Introduction

The computation of detailed stellar stellar spectra is a basic but important step required for the analysis of star light, spectra of integrated stellar populations, or transiting planets. The calculations involve solving the equation of radiative transfer

$$\cos\theta \frac{dI_\nu(\theta)}{dz} = \eta_\nu - \kappa_\nu I_\nu(\theta) + \int_0^\pi I_\nu(\theta')\phi(\theta')\sin\theta' d\theta', \quad (1)$$

where I_ν is the specific intensity, and describes the radiation field (radiative energy per unit of frequency propagating at an angle θ from the vertical traversing a unit area per unit time), κ is the opacity (fraction of energy absorbed per unit length), η the emissivity (energy emited, same units as the intensity) and ϕ represents the scattering function (photons scattered from the direction θ' into the direction θ.

Once I_ν (θ) is known, it can be integrated to derive the stellar flux

$$H_\nu = \int_0^\pi I_\nu(\theta)\cos\theta\sin\theta d\theta \quad (2)$$

which can be finally convolved with various kernels that describe large-scale atmospheric turbulence, rotation, or the instrumental profile.

Prior to solving the equation of radiative transfer, one requires a physical model of the stellar atmosphere in order to compute η, κ and ϕ. This is the running of the thermodynamical quantities (e.g temperature and density) with height, as well as the chemical composition. This input *model atmosphere* is the sole required input to Synple. Of course, computing a detailed spectrum requires fundamental physical data on the interaction of matter and radiation (photoionization cross-sections, atomic and molecular transition probabilities and damping constants, etc), but a generic collection of such data is bundled with Synspec.

Multiple codes for radiative transfer exist and are publicly available. However, most require deep knowledge of the related physics and are cumbersome to use. Synple provides an easy-to-use Python interface to Synspec (Hubeny & Lanz 2017; Hubeny et al. 2021) for the fast computation of model stellar spectra with accuracy and flexibility.

2 Installing

Synple is hosted in github at https://github.com/callendeprieto/synple and can be obtained using `git`
`git clone https://github.com/callendeprieto/synple synple`
or by downloading a zip file from the website.

Your Python (3.x) installation will require the packages `subprocess`, `numpy`, and `scipy`, usually obtained with `pip` (e.g. `pip3 install numpy`). You probably want to install ipython and matplotlib as well, since we will be using that in the examples below.

The installation requires a FORTRAN compiler, and the GNU compiler is used by default
`cd synple/synspec ; make clean; make; cd ../..`

The atomic and molecular line lists are sizable and need to be downloaded separately
`cd synple/linelists ; make clean; make; cd ../..`
where the automated download requires that `wget` is available in your system, otherwise download the line lists manually from
`ftp://carlos:allende@ftp.ll.iac.es/linelists` and copy them to the synple/linelists directory. For speed the linelists are converted to binary format by the same (make) script.

You need to make synple visible to Python by including the synple directory in your PYTHONPATH variable. Say you've placed the synple parent directory in your home, then you can add it in the linux shell by typing (or including in your .bashrc or .cshrc files)

bash: export PYTHONPATH="$PYTHONPATH:$HOME/synple"
csh: setenv PYTHONPATH = "$PYTHONPATH:$HOME/synple" .

3 Examples

Once synple is installed, all you need is a model atmosphere to being using it. In this section we'll include a set of simple practical examples, mainly based on the three model atmospheres included in the synple/models directory: one is a solar Kurucz model ('ksun.mod'; Kurucz 1979 and later updates), one is

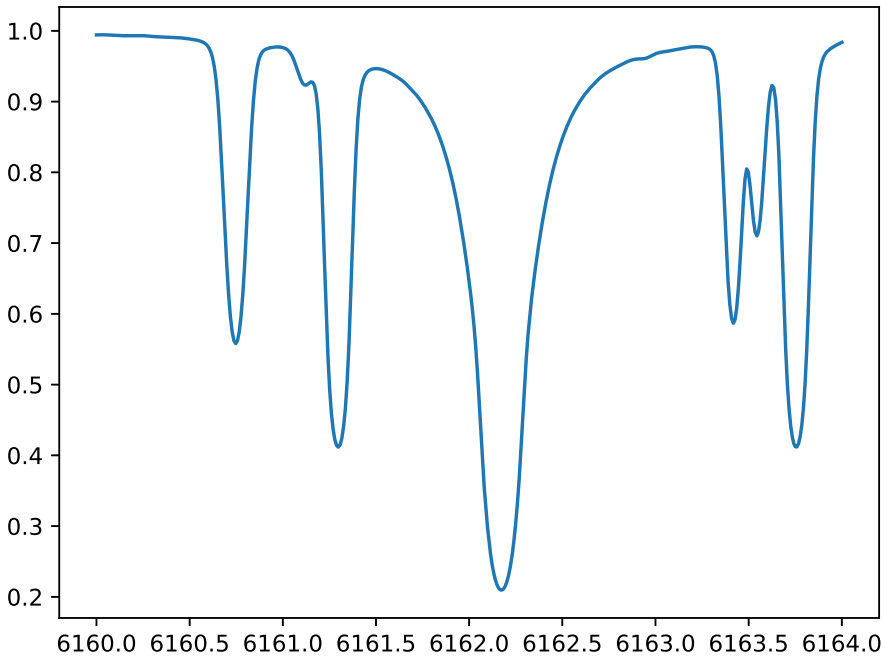

Figure 1: Here's your first spectrum with synple. You have modeled the solar spectrum in the vicinity of the Ca I λ6162 line. The Ca I line is the strong line at the center, visibly damped by collisions with hydrogen atoms.

a solar MARCS model ('msun.mod'; Gustafsson et al. 2008), and the third is a Phoenix model for Vega-like A-type star ('lte09600-4.00-0.0.PHOENIX-ACES-AGSS-COND-2011.ATMOS.fits') from Husser et al. (2013).

3.1 Computing your first solar spectrum

For this exercise we'll use the Kurucz solar model included with synple. We're going to compute a piece of the spectrum around the Ca I λ6162 line. We start ipython and import the standard synthesis routine `syn` from synple

```
$ ipython
Python 3.6.8 (default, Jan 14 2019, 11:02:34)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.4.0 -- An enhanced Interactive Python.  Type '?'  for help.

In [1]:  from synple import syn
```

then request the computation of the spectrum from the model atmosphere in the file *ksun.mod* between 6160 and 6164 Å

```
In [2]:  wave, flux, cont = syn('ksun.mod', (6160,6164) )
teff,logg,vmicro= 5777.0 4.437 2.0
syn ellapsed time 48.8003294467926 seconds
```

We've got it! The routine `syn` returns three numpy arrays with the wavelengths (wave), the fluxes (flux; H_λ), and the continuum flux (cont). If we wish to visualize the continuum-normalized spectrum, we can use matplotlib

```
In [3]:  pylab
Using matplotlib backend:  TkAgg
```

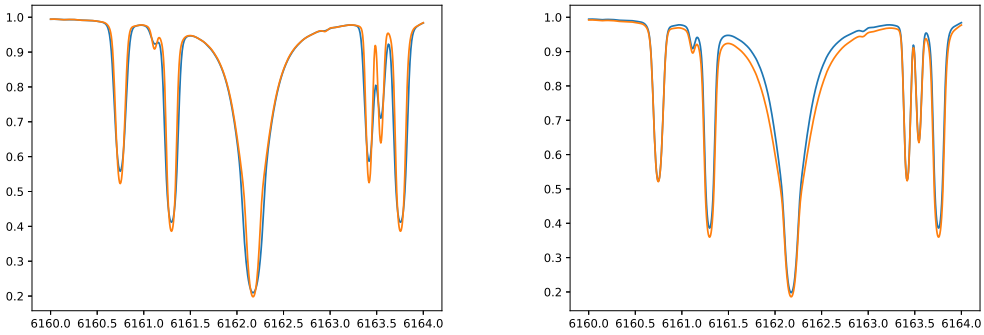

Figure 2: Repeating the calculation in Fig 3.1 (blue) but adding a second curve (orange) on the left panel reducing the micro-turbulence from 2 to 1 km s^{-1}, and on the right-hand panel increasen the Ca abundance by 0.2 dex.

```
Populating the interactive namespace from numpy and matplotlib

In [4]:  plot(wave,flux/cont)
```

and that's it! You should be looking at your first computed spectrum and it should look like the one in Fig. 1. In addition to the run with height of the thermodyanical quantities, the model atmosphere contains a detailed description of the chemical abundances and the micro-turbulence velocity (see below). These parameters are simply adopted from the model atmosphere, but can be changed if needed, as described in the next section.

3.2 Altering micro-turbulence and chemical composition

The model atmospheres we deal with are in hydrostatic equilibrium, and therefore pressure and gravity are in balance, so everything is at rest. This is unrealistic. Real stellar atmospheres have turbulence, shocks, convection, meridional circulation, etc. Thus, the computed line profiles are narrower than observed. One of the patches to address this problem is the so-called micro-turbulence velocity, which accounts for small-scale velocities of the absorbing particles. This includes small scale turbulence, and needs to be included in the computation of the linel absorption profile at the microscopic level, before solving the radiative transfer equation.

Synple includes micro-turbulence in the spectrum calculations automatically, if that's indicated in the model atmosphere file, but that value can be overridden by explicitly indicating it when calling the synthesis routine `syn` using the parameter *vmicro*.

Let's repeat the example in §3.1 changing the micro-turbulence from the value used to construct the solar Kurucz model we used, 2 km s^{-1}, to a more reasonable value for a solar-like star of 1 km s^{-1}

```
In [5]:  wave2, flux2, cont2 = syn('ksun.mod', (6160,6164), vmicro = 1.  )
teff,logg,vmicro= 5777.0 4.437 1.0
```

which can be easily compared with the previous calculation

```
In [6]:  plot(wave2,flux2/cont2)
```

as illustrated in the left-hand panel of Fig. 2. The weak, unsaturated, lines, and the core of the strong Ca I λ6162 line, are less broadened (brown curve) after reducing the micro-turbulence, while the damping wings of strong lines are insensitive to this parameter.

Similarly to the micro-turbulence, the chemical abundances can be changed from those used in the construction of the model atmosphere when computing the detailed spectrum. Since the most direct effect of the change of the abundance of an element (other than hydrogen) in the atmosphere is on the strength of its spectral lines, this is usually a good approximation for most elements and for reasonably small changes in abundances. But many elements, especially the most abundant ones, affect the thermodynamics of the atmosphere, and therefore for those elements this is a risky approximation to make. On the other hand, the approximation is very good for trace elements.

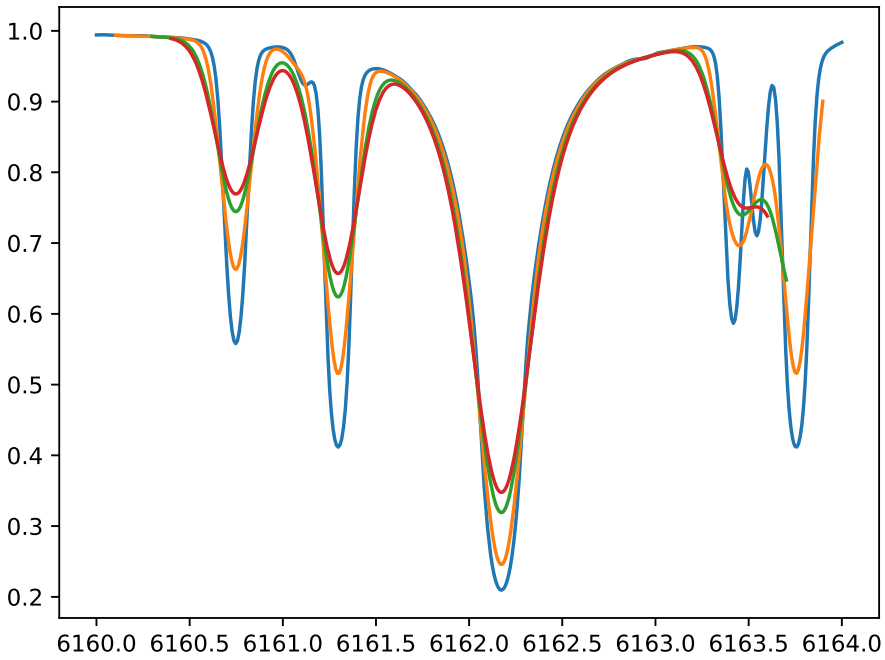

Figure 3: Repeating the calculation in Fig 3.1 (blue) but adding the result of a convolution with a rotational profile for $v \sin i = 5$ km s^{-1} (orange), a Gaussian kernel with a FWHM of 0.2 Å (green), and both (brown).

To supersede the abundances in the model atmosphere we can call `syn` with the keyword *abu*. The abundances are expressed as the number density of nucleii of the elements relative to that of hydrogen

$$\epsilon(X) = \frac{N(X)}{N(H)}. \tag{3}$$

Let's read the ones in the model atmosphere and then increase the calcium abundance by 50% (0.2 dex)

```
In [7]:  from synple import read_model
In [8]:  atmostype, teff, logg, vmicro, abu, nd, atmos = read_model('ksun.mod')
In [9]:  abu[19] = abu[19] * 1.5
In [10]:  wave3, flux3, cont3 = syn('ksun.mod', (6160,6164), vmicro = 1.  , abu=abu )
teff,logg,vmicro= 5777.0 4.437 1.0.
In [11]:  clf()
In [12]:  plot(wave2,flux2/cont2, wave3, flux3/cont3).
```

This calculation is shown in the right-hand panel of Fig. 2, and it reveals that there are three Ca I lines in the computed spectral segment.

3.3 Additional line broadening

The spectral lines in a computed spectrum are naturally broadened due to the uncertainty principle, micro-turbulence (see Section 3.2), the thermal velocities of the absorbing atoms and molecules, and the collisions they suffer with surrounding free electrons and hydrogen atoms. All these broadening factors are included in the calculations done as described in the previous exercise, but there are additional sources of uncertainty that may be significant for your particular application.

- macro-turbulence: while the *micro* takes care of small-scale velocity fields by introducing broadening at the microscopic level, broadening the line profiles at each atmospheric layer, there are velocities on scales much larger than the photon mean free-path that need to be taken into account by convolution of the computed spectrum. This is usually taken into account by convolving with a

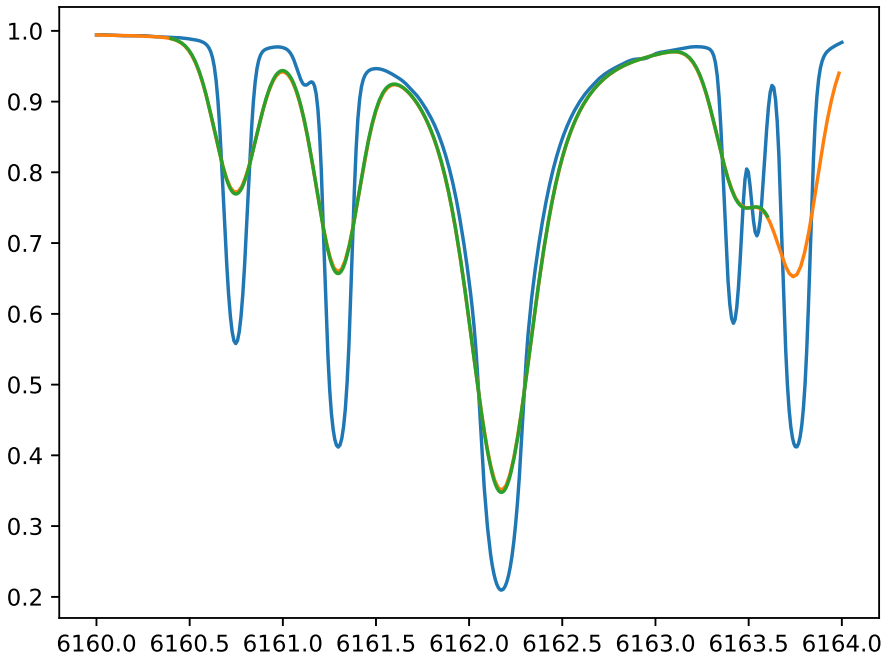

Figure 4: In addition to the original calculation in Fig. 3.1 (blue line), we show the result of the convolution with a rotational profile for $v \sin i = 5$ km s^{-1} and a Gaussian kernel with a FWHM of 0.2 Å using `rotconvlgconv` (green) and rotin (orange).

Gaussian velocity kernel, or with a radian-tangencial profile (see Gray xxxx). Synple implements convolution with a Gaussian velocity field in the routine `vgconv`.

- rotation: stellar rotation broadens the spectral profiles. To be exact this has to be taken into account numerically, by Doppler-shifting the intensity contributions from different angles (θ) in Eq. 2. Nevertheless the effect or rotation can be well approximated by a covolution of the flux computed for no rotation with a particular kernel (see Gray xxxx). This is implemented in Synple in the routine `rotconv`.
- instrumental profile: Instruments distort the stellar spectrum by convolving it with the instrumental profile. If the resolving power is constant, this can be usually approximated as a macro-turbulence, with a Gaussian kernel with a constant width in velocity space. Sometimes is the resolution, the FWHM in wavelength of the response of the instrument to a monochromatic source, what is constant, and the instrumental profile can be handle in Synple with the routine `lgconv`.

The following example demonstrates how to convolve with a rotational profile ($vsini = 5.$ km s^{-1}), a Gaussian profile with a constant resolution FWHM = 0.2 Å and both, the original calculation in §3.1

```
In [13]:  from synple import rotconv, lgconv
In [14]:  wave2, flux2 = rotconv(wave, flux/cont, 5.)
In [15]:  wave3, flux3 = lgconv(wave, flux/cont, 0.2)
In [16]:  wave4, flux4 = lgconv(wave2, flux2, 0.2)
In [17]:  clf()
In [18]:  plot(wave, flux/cont, wave2,flux2, wave3, flux3, wave4, flux4).
```

Note that the spectral range of the output spectrum is truncated in wavelength, since it is computed only for the frequencies with information in the original calculation.

These routines for convolution are directly implemented in Python. Synspec has a companion FORTRAN program that handles the convolution with the instrumental or rotational profiles: rotin. One can use this program by including the *fwhm* and *vrot* parameters when calling `syn` as in the following example

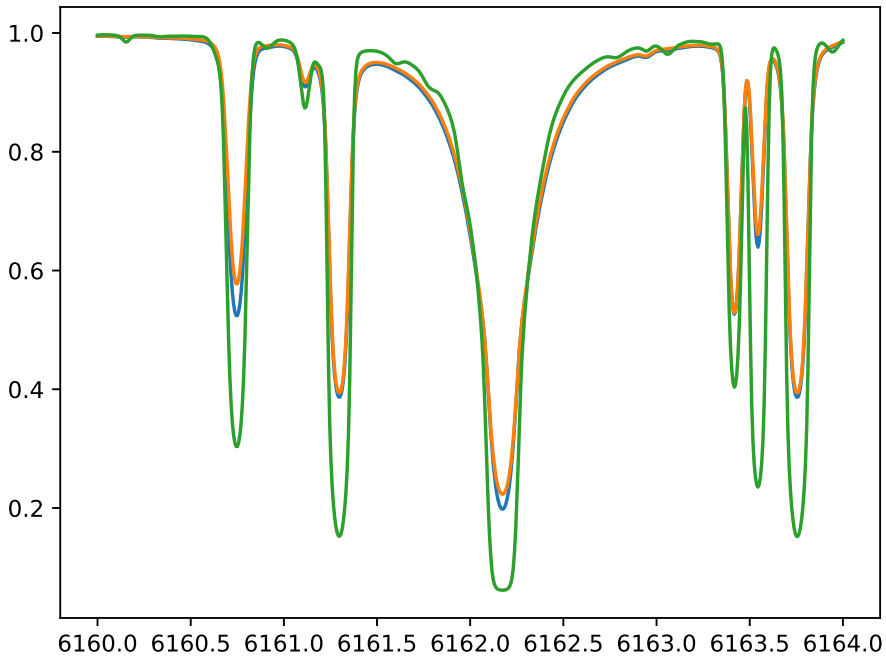

Figure 5: We now use all the models in the 'modeldir' directory in one command, with multiple values of the micro, using `multisyn`.

```
In [19]:  wave2, flux2, cont2 = syn('ksun.mod', (6160,6164) , vrot = 5.0, fwhm = 0.2 )
In [20]:  clf()
In [21]:  plot(wave, flux/cont, wave2, flux2/cont2, wave4, flux4)
```

Note the different behaviour for rotin (orange curve in Fig. 5), which does not trim the edges of the computed spectrum, and `lgconv`/`rotconv` (green). Rotin does not include convolution with a Gaussian kernel with a constant width in velocity, implemented in `vgconv`.

3.4 Multiple models

One often faces the need to compute the spectra for multiple models. Since the sampling of frequencies is internally controlled within synspec, it becomes desirable to force the output fluxes to be on the same wavelength grid, and this can be done using the parameter *dw* in the routine `syn`.

When using synple, the default value for the variable `modeldir` points to the folder `models` inside you `synple` parent directory. If the model atmospheres are in your working directory, or as in the first examples below, the models are in the `models` folder, that's ok, but if you want to use models elsewhere you will need to change `modeldir` to point to the right path (changing the source code in synple.py) or simply passing the complete filenames, including an absolute path.

If you have a bunch of models you want to compute spectra for, or multiple values of vrot, fwhm, or vmicro, you can wrap around the routine syn, or you can use the macro `multisyn`, after setting the file names in a list like in this example

```
In [22]:  from synple import multisyn
In [23]:  import glob
In [24]:  import os.path
In [25]:  modeldir = "/home/callende/synple/models"
In [26]:  models = glob.glob(os.path.join(modeldir,"*mod")) #get the models in a list
In [27]:  wave5, flux5, cont5 = multisyn(models, (6160,6164), vmicro = [1.,2.,4.]  )
In [28]:  clf()
In [29]:  for i in range(len(models)):  plot(wave5, flux5[i,:]/cont5[i,:])
```

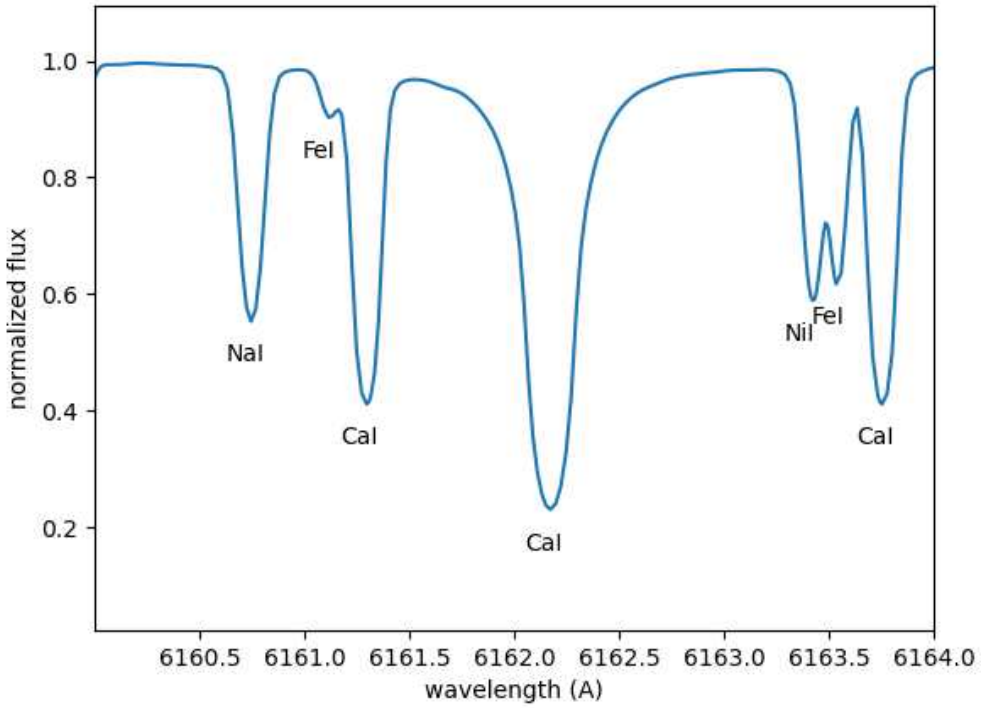

Figure 6: The main transitions in the spectrum can be labeled activating the `tag` keyword in `syn`.

and the wavelength array will now be common to all spectra, and the flux and cont arrays returned will be two-dimensional, with the first dimention running along the models in the input list.

4 Identification of spectral features

Synple can be used to identify which transitions are causing any given spectral feature. Of course, to succeed you need to use a line list including the relevant transitions, with data of sufficient accuracy.

As we have seen, the usual output from the routine `syn` is a tuple with three arrays: wavelengths, fluxes and continuum fluxes. When the parameter `tag` is set to **True** in `syn`, the output changes including a 4-th element, a list with three additional arrays that give

1. the wavelengths of the most relevant transitions,
2. the ions or molecules responsible for those transitions, and
3. the estimated equivalent widths associated with them (in milliangstroms).

For example, for the very first spectrum computed in section 3.1, we could have used this feature

```
In [1]:  from synple import syn, tags

In [2]:  d = syn('ksun.mod', (6160,6164) , tag = True)
```

to produce the output graphics shown in Figure 6. The routine `tags` can be used to plot the output from syn for transitions expected to be stronger than an specified threshold in equivalent width, e.g.

```
In [3]:  tags(d, 1.)
```

will tag several several additional features estimated to produce absorptions stronger than 1 mÅ, and not included when the default minimum of 10 mÅ is used.

5 Grid handling

Synple includes a number of tools for creating grids of synthetic spectra. The format adopted is the same used for the FERRE grids, described in Section 4 of the FERRE manual available from
`https://github.com/callendeprieto/ferre/blob/master/docs/ferre.pdf`

Here is a quick overview of the various tools.

- head_synth(synthfile): extracts the header of a grid
- lambda_synth(synthfile): extracts the wavelength array
- read_synth(synthfile,nd=False): reads the header, parameters and fluxes from a file
- write_synth(synthfile,d,hdr=None): writes a grid to file
- fill_synth(d,kernel='thin_plate_spline',neighbors=100): fills-in missing data using Radial Basis Functions (RBF) interpolation
- rbf_get(synthfile, kernel='thin_plate_spline'): finds the coefficients for RBF interpolation
- rbf_apply(synthfile,c,par): applies the RBF coefficients to interpolate
- gsynth(synthfile,...): applies smoothing and reddening to an existing grid
- polysyn(modelfiles, wrange, ...): sets up a directory tree for computing spectra for a grid using a job scheduler such as slurm
- collect_marcs(modeldir=modeldir, tteff=None, ...): collects MARCS model atmospheres for grid computation
- collect_kurucz(modeldir=modeldir, tteff=None, ...): collects Kurucz model atmospheres for grid computation
- collect_k2odfnew(modeldir=modeldir, ...): collects Kurucz ODFNEW model atmospheres for grid computation
- mkgrid(synthfile=None, tteff=None, ...): collects the results from a calculation prepared using polysyn for a regular grid and stores the spectra in FERRE format
- mkgrid_irregular(synthfile=None, tteff=None, ...): collects the results from a calculation prepared using polysyn for an irregular grid and stores the spectra in FERRE format.

6 Computing opacity tables

Synple provides simplified access to the most basic features for opacity table calculations available with synspec. The main subroutines are

- polyopt(wrange=(9.e2,1.e5), dlw=2.1e-5, ...): sets up an opacity grid calculation
- read_opt(filename): reads an opacity table
- read_copt(filename,nrho,nt): reads the continuum opacities from an opacity grid calculation.

For example, to compute a minimalist table covering 10 Å (6100. -6110. Å), 2 temperatures (3163 and 5012 K), and 2 densities (1e-14, 1e-13 gr cm^{-3}):

```
In [1]:  from synple import polyopt
In [2]:  polyopt(wrange=(6100.,6110.),tlt=(2,3.5,0.2),tlrho=(2,-14.,1.0))
```

The code creates 1 folder named `hyd0000001` (you would get more folders by including more chemical mixtures, but the default is to assume solar abundances) with a script *hyd0000001.job*. Executing the script should produce an output text file **opt.data** with the opacity table.

To read and plot the table

```
In [1]:  from synple import read_opt
In [2]:  import matplotlib.pyplot as plt
```

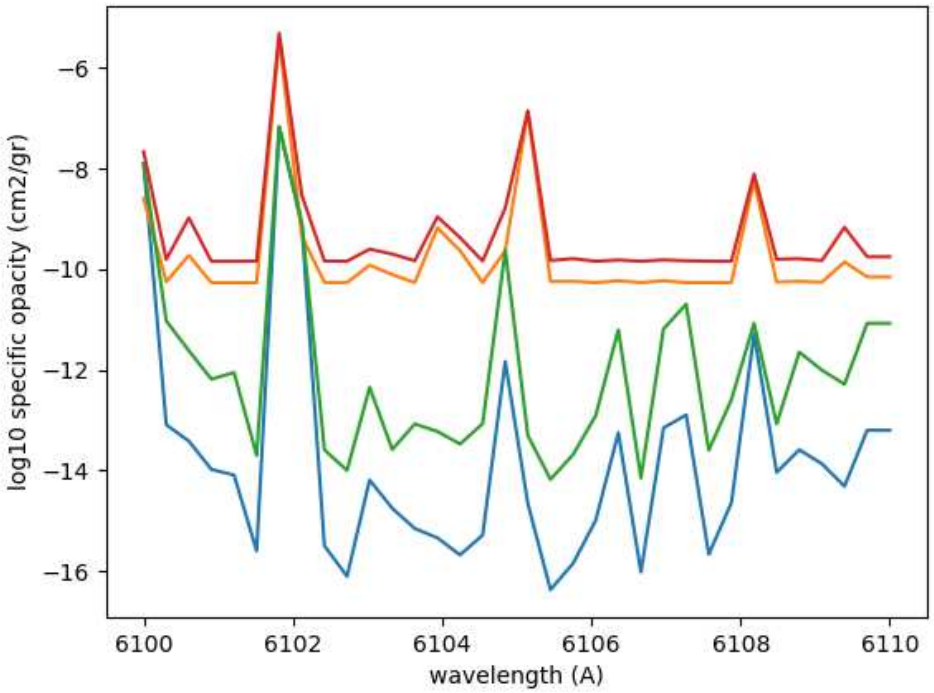

Figure 7: Opacity in the range 6100–6110. Å for four combinations of a pair of densities ($\log_{10}\rho = -32.236191, -29.933606$) and temperatures ($\log_{10} T = 8.059055, 8.51957$) and solar abundances.

```
In [3]:  lrho,lt,lambda0,lopa,abu_eos,abu_opa = read_opt('opt.data')
In [4]:  plt.plot(lambda0,lopa[:,0,0])
In [5]:  plt.plot(lambda0,lopa[:,0,1])
In [6]:  plt.plot(lambda0,lopa[:,1,0])
In [7]:  plt.plot(lambda0,lopa[:,1,1])
In [8]:  plt.xlabel('wavelength (A)')
In [9]:  plt.ylabel('log10 specific opacity (cm2/gr)')
In [10]:  plt.show().
```

which will produce the output shown in Figure 7.

7 Computing emergent intensities I_ν

It is very easy to tweak the basic routine *syn* (see Sect. 3.1), or the parallel versions *polysyn* or *raysyn*, to output the specific intensity as a function of the angle from the vertical (θ). One simply needs to set the keyword `intensity` to true, as in the following example. When this is activated the output from `syn` will change from a tuple with 3 arrays (4 if `tag` is set, as explained in Sect. 4) to a tuple with 4 arrays (5 if `tag` is set), and the 4th array will be a 2D matrix with the emergent intensity for 10 inclinations corresponding to $\mu \equiv= \cos\theta = 0.1, 0.2, \ldots, 1.0$.

```
In [1]:  from synple import syn
In [2]:  import matplotlib.pyplot as plt
In [3]:  wave, flux, con, inte = syn('ksun.mod', (6160,6164), intensity=True )
/home/callende/synple/models/ksun.mod is a kurucz model
teff,logg,vmicro= 5777.0 4.437 2.0
syn ellapsed time 16.27164101600647 seconds

In [4]:  for angle in range(10):  plt.plot(wave,inte[:,angle])
In [5]:  for angle in range(10):  plt.text(6160.2,max(inte[:,angle]),'μ ='+"0.
In [6]:  plt.ylabel('I$_{nu}$ (erg/s/cm2/Hz)')
```

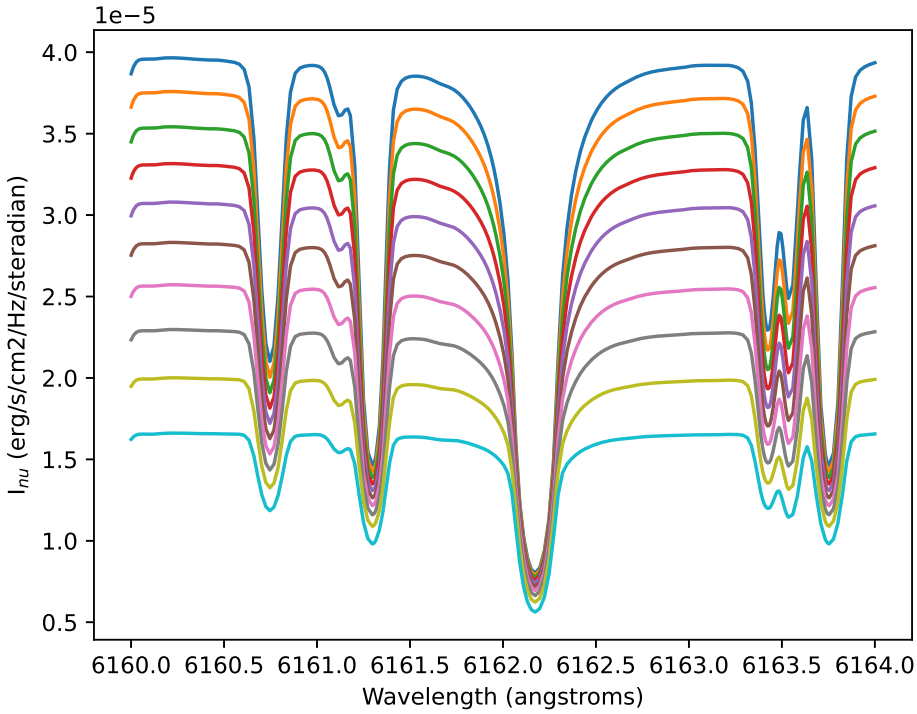

Figure 8: Example of the computation of emergent specific intensities I_ν for various directions. The units are in erg cm^2 s^2 Hz^{-1} steradian^{-1}.

```
In [7]:  plt.xlabel('Wavelength (angstroms)')
In [8]:  plt.show()
```

which will produce the graph shown in Fig. 8. Note that this option is not compatible with the broadening parameters `fwhm`, `vrot` or `vmacro`.

8 References

- Allende Prieto, C., Fernández-Alvar, E., Schlesinger, K. J., et al. 2014, A&A, 568, A7
- Gustafsson, B., Edvardsson, B., Eriksson, K., et al. 2008, A&A, 486, 951
- Hubeny, I. & Lanz, T. 2017, arXiv:1706.01859
- Hubeny, I., Allende Prieto, C., Osorio, Y., et al. 2021, arXiv:2104.02829
- Husser, T.-O., Wende-von Berg, S., Dreizler, S., et al. 2013, A&A, 553, A6
- Kurucz, R. L. 1979, ApJS, 40, 1