ISPy3 - Integrated-light Spectroscopy with Python 3

Søren S. Larsen

July 10, 2020

Contents

1 Introduction

ISPy3 is a collection of Python routines that can be used to model the integrated-light spectra of stellar populations. The actual spectral synthesis and related tasks (setting up model atmospheres, etc) is done via calls to external codes. Currently, the Kurucz codes (ATLAS/SYNTHE) and MARCS/TurboSpectrum are supported, although it should be relatively straight forward to implement other similar codes. As such, ISPy3 requires these external codes as well as their required input data (atomic and molecular line lists, opacity distribution files, etc) to be installed in addition to the Python files.

The original motivation for developing ISPy3 was to determine chemical abundances from integrated-light spectra of globular clusters, and the package includes high-level routines that can be used to determine the abundances of specified elements via least-squares fits to observed spectra. However, the package can also be used more generally to compute model atmospheres and synthetic spectra via Python wrapper routines for the model atmosphere and spectral synthesis codes. In addition, ISPy3 includes utility functions for a variety of other purposes, such as computing equivalent widths of individual spectral lines (using a modified version of the Kurucz WIDTH9 programme) and generating opacity distribution functions (ODFs) for ATLAS9.

The emphasis in the present document is on practical aspects of setting up and running the Python code. The ATLAS/SYNTHE and MARCS/TurboSpectrum models and codes are documented elsewhere, and that documentation should be consulted for more information. This document is *not* intended as a detailed guide to these external codes and their various features, caveats and idiosyncracies. An earlier version of ISPy3 was introduced in Larsen et al. (2012), which may be consulted for an example of a practical use case, as well as a more in-depth discussion of the input data and a number of tests of the performance of the code. Additional references are Hernandez et al. (2017, 2018) and Larsen et al. (2014, 2017). Before proceeding it is worth repeating the warning given on the Kurucz website, which also applies here:

> *Neither the programs nor data are "black boxes". You should not be using them if you do not have some understanding of the physics and of the programming in the source code.*

2 Installation

The ISPy3 Python tools require a basic Python 3 installation, including the numpy and scipy packages. To install the Python files, simply download and untar the ispy3.tar.gz file:

```
>> tar xvzf ispy3.tar.gz
```

and make sure that the directory in which the package is installed is included in your Python search path. The Python files will not by themselves be very useful without the external executables, which ISPy3 expects to find in the directory defined by the absetup.binpath variable, and their associated input data (line lists, ODFs, etc.)

Users are warned that ISPy3 will generally trust that the ancillary data files and external executables are present in the correct format in the expected locations. If this is not the case it will typically crash ungracefully, without necessarily producing very informative error messages.

3 General overview and prerequisites

Neither the Kurucz nor Turbospectrum codes make use of modern multi-core architectures, but the problem of computing integrated-light spectra nevertheless lends itself well to making use of parallel processing capabilities. In `ISPy3`, an integrated-light model spectrum is computed by dividing the underlying Hertzsprung-Russell Diagram (HRD) of the population into a number of bins (typically 50-100) and computing model atmospheres and synthetic spectra for each bin. The individual model spectra are then co-added:

$$L_{\mathrm{SSP}}(\lambda) = \sum_{i=1}^{n} w_i L_i(\lambda) \tag{1}$$

where $L_{\mathrm{SSP}}(\lambda)$ is the luminosity of the resulting "simple stellar population" (SSP) at wavelength λ, $L_i(\lambda)$ the luminosity of a star in the ith bin, and w_i a specified set of weights. The weights are typically given by the stellar mass function $\xi(M) \equiv \mathrm{d}N/\mathrm{d}M$:

$$w_i = \xi(M_i)\Delta M_i \tag{2}$$

for a mass bin covering a range ΔM_i of stellar masses. As each bin is independent of the others, the individual model spectra $L_i(\lambda)$ can be computed in parallel, the only limitation on the number of parallel processes being set by the available number of CPU cores and disc space for storage of temporary files. In principle, memory might also be a limitation but in practice neither of these codes are particularly memory intensive by modern standards.

In general, the Kurucz `SYNTHE` spectral synthesis code is used together with `ATLAS` model atmospheres, while `TurboSpectrum` is used with `MARCS` model atmospheres (the most recent version of `TurboSpectrum`, 2019, can also read `ATLAS` model atmospheres). Both combinations have pros and cons: The `ATLAS` atmosphere models are computed via calls to the actual `ATLAS9` or `ATLAS12` codes, and can thus in principle be computed for any desired set of physical parameters (composition, gravity, temperature, etc). Various details of the models can be customised, such as the number of atmospheric layers, the range of optical depths, and microturbulence. A drawback is that the `ATLAS` models only support plane-parallel geometry. The `MARCS` models are available for spherical geometry, but as the `MARCS` code is not publically available the parameter range is restricted to what can be obtained by interpolation in the precomputed model grid available via the `MARCS` website. It should be emphasised that the computation of model atmospheres and the corresponding synthetic spectra are separate steps; both `SYNTHE` and `TurboSpectrum` can compute model spectra for compositions that do not necessarily match those of the atmospheres in detail. In fact, there is nothing to stop the adventurous user from computing a model spectrum with a completely different composition than that of the model atmosphere, although this is not physically consistent and the usefulness of such an experiment may be rather questionable.

Apart from these and various other technical differences in the detailed implementations of the codes, the atmosphere and spectral synthesis models supported by `ISPy3` are classical models with the usual standard assumptions: The geometry is one-dimensional (i.e. the only spatial variable is the depth), the models are plane-parallel and static, and they assume Local Thermodynamic Equilibrium (LTE). As any good textbook on stellar atmospheres will stress, none of these assumptions are physically realistic, but they are nevertheless very convenient computationally, and make it feasible to compute large numbers of models in a reasonable amount of time.

3.1 The Kurucz codes: ATLAS, SYNTHE, etc.

As noted above, the ATLAS models come in two flavours. ATLAS9 uses pre-defined opacity distribution functions (ODF) to calculate the opacities, whereas ATLAS12 calculates opacities via direct opacity sampling (OS). The OS method allows the calculation of atmosphere models for any specified composition, but is computationally much more expensive. In addition, it can sometimes be difficult to get ATLAS12 models to converge, especially for high surface gravity models with low temperatures ($T_{\rm eff} < 4000$ K). The ATLAS9 models are much faster to compute and tend to converge more easily, but are restricted to compositions for which ODFs are available. In ISPy3, ODFs are included for Solar-scaled and α-enhanced composition and metallicities $-3.50 \leq$ [Fe/H] $\leq +0.50$, but ODFs for other compositions can be computed with the `kurucz.calcodf()` function.

ISPy3 computes ATLAS models for any specified combination of physical parameters ($T_{\rm eff}$, $\log g$, composition) by calling the corresponding Kurucz codes. An ATLAS9 model can be computed from scratch, or starting from a pre-existing model with physical parameters close to the desired ones. The default behaviour is to look for the closest matching model among those available in the pre-computed set by Castelli and use that as a starting guess. Alternatively, an initial model can be specified directly when calling the `kurucz.mkatm()` function. The recommended procedure is to start from a pre-existing model, as this usually ensures faster convergence. An ATLAS12 model must always be computed starting from a pre-existing model, usually computed with ATLAS9. Again, the default behaviour is to look for a Castelli model, but a starting model can also be specified directly when calling `kurucz.mkatm_a12()`.

The final computed models will be stored with a filename ending with `.A9` or `.A12`. In addition, a file ending with `.out` is produced, containing diagnostic output from the ATLAS9 or ATLAS12 codes. At the end of this file, some extra information about each layer in the model is listed, with the last two columns (the errors on the flux and on the flux derivative) being useful for assessing whether the model has properly converged. If this information is not found at the end of the `.out` file, it means that the procedure crashed for some reason.

Spectral synthesis based on ATLAS models is normally done with the SYNTHE code. Assuming that an ATLAS (9 or 12) model is available, the first step is to call `syntherk.synbeg()`. This will define the wavelength range and spectral resolution, as well as the elements and molecules to include in the model spectrum, whether the predicted lines should be included, etc. It is possible to specify explicitly which molecules and elements to include in the spectral synthesis; the default is to include everything. Line data for the specified wavelength range will then be read and stored in temporary files. `syntherk.synbeg()` will produce two versions of these files, with and without the TiO lines as these can be skipped for temperatures higher than 4500 K, which makes the calculations much faster. The actual model spectrum is then calculated with a call to `syntherk.synthespec()` which automatically selects the appropriate input files, depending on the temperature.

3.1.1 Obtaining and installing the Kurucz codes

The Kurucz suite includes quite a large number of programmes to carry out various tasks related to the computation of model atmospheres and synthetic spectra. The primary reference for the source codes, line data, etc., is Kurucz' website at `http://kurucz.harvard.edu` where the original (VMS) versions of the Fortran codes can be found. Versions of the source code suitable for compilation under Linux and macOS are available at the websites of F. Castelli: `http://wwwuser.oats.inaf.it/castelli/` and Observatoire de Paris `http://atmos.obspm.fr`. The latter are documented in Sbordone (2005). The Intel Fortran compiler (`ifort`) is required to compile the Fortran source code; unfortunately this compiler is no longer free, but precompiled binaries for Linux are available from the Observatoire de Paris site, which also has a Makefile that makes it straight forward to compile the codes. The Castelli website has versions of the codes that can be compiled with `gfortran`, but I haven't tested these extensively. They do seem to run somewhat slower, but this might have more to do with the different compiler than with the codes themselves.

Once the codes have been compiled, they should be copied to the directory `absetup.binpath`, where `ISPy3` expects to find them. Below is an overview of the various Kurucz codes.

3.1.2 Kurucz codes for calculating model atmospheres

atlas12.exe `ATLAS12` code for calculation of model atmospheres for arbitrary composition and microturbulence with the opacity sampling method. I am using a slightly modified version of the code than that distributed by Sbordone / Castelli, in which the maximum number of atmospheric layers has been increased to 99 (instead of 72) and the TCORR subroutine has been slightly modified so that it is similar to the one in `atlas9mem.for`. The latter seems to alleviate the convergence problems with cool, high surface gravity atmospheres in some cases.

atlas9mem.exe `ATLAS9` code for calculation of model atmospheres with pre-computed Opacity Distribution Functions.

kappa9.exe Computes Rosseland mean opacities from ODFs.

kapreadts.exe Rearranges output from `kappa9.exe` for `ATLAS9`.

dfinterpbig.exe Interpolates in pre-computed ODFs to make an ODF for the appropriate metallicity.

3.1.3 Kurucz codes for spectral synthesis

rgfalllinesnew.exe Read atomic line data.

rmolecasc.exe Read molecular line data from ASCII files. The most recent version of this programme, available from the Kurucz web site, is required in order to read data for MgO, CaO, NaH, KH, CaH, and CrH. However, the Kurucz version needs to be slightly modified to write the data in a format that is compatible with the Castelli/Sbordone distributions of

SYNTHE (the output must include the 'alpha' exponent for the ABO-style van der Waals damping, even though alpha is always 0 for molecular data).

rpredict.exe Read data for predicted lines.

rh2ofast.exe Read binary H_2O line data.

rschwenk.exe Read binary TiO line data (Schwenke).

xnfpelsyn.exe Precomputes continuum opacities and number densities for various elements.

synbeg.exe Initialises the spectral synthesis procedure - defines wavelength range and spectral resolution.

synthe.exe Calculates the line opacity data for each line in the spectrum.

spectrv.exe Calculates continuum opacities and final synthetic spectrum.

rotate.exe Adds rotational broadening to the computed spectrum (optional).

converfsynnmtoa.exe Converts the binary output files (from `spectrv.exe` or `rotate.exe`) to ASCII format.

3.1.4 Kurucz codes required for calculation of equivalent widths

The WIDTH9 code is quite versatile and can be used in several different ways, depending on the input cards provided. ISPy3 uses the `inpwidth.for` code written by F. Castelli to produce the input files needed by WIDTH9 to compute equivalent widths, consisting of the control cards (including a model atmosphere) and line list in a modified format.

inpw_sl.exe Slightly modified version of Castelli's `inpwidth.for` code, which only needs a single line list as input and allows the wavelength matching tolerance for line identification to be specified by the user.

width9sl.exe Modified version of the Kurucz WIDTH9 code. The modifications are as follows: 1) a curve of growth can be computed for up to 99 abundance steps (rather than the standard 9), 2) The continuum flux will be written to the output file (needed by `abutils.hrd2ew()`), 3) The new VOIGT subroutine by Castelli has been replaced by the original Kurucz subroutine.

3.1.5 Kurucz codes required for calculation of ODFs

xnfdf.exe

dfsynthe.exe

dfsortp.exe

separatedf.exe

3.1.6 Input files for Kurucz codes

Line lists for SYNTHE are available at the Kurucz website, where descriptions of the formats can also be found. The locations of the line lists are defined in the `syntherk.py` module in the variables `syntherk.atomdir` (for the atomic lines) and `syntherk.moldir` (for the molecular lines). There is a separate directory for the predicted lines, `syntherk.preddir`, which are only included in the calculations if specifically requested by setting the argument `predicted=True` when calling `syntherk.synbeg()`. The line lists are, of course, a critical ingredient for the modelling of any spectrum, and the raw line lists from the Kurucz website are far from perfect.

ISPy3 expects the full atomic line list to be split up into smaller chunks. These files can be produced from the full line list with the programme `splitgf100.for`:

gf0150.100 : Wavelengths < 1500 Å

gf0200.100 : 1500 Å < λ < 2000 Å

gf0300.100 : 2000 Å < λ < 3000 Å

gf0400.100 : 3000 Å < λ < 4000 Å

gf0500.100 : 4000 Å < λ < 5000 Å

gf0600.100 : 5000 Å < λ < 6000 Å

gf0800.100 : 6000 Å < λ < 8000 Å

gf1200.100 : 8000 Å < λ < 12000 Å

gf3000.100 : 12000 Å < λ < 30000 Å

SYNTHE supports the use of "ABO" (Anstee, Barklem, & O'Mara) damping constants for atomic lines, although these are not included in the standard line lists available at the Kurucz website and there is little documentation for this feature.

The molecules listed below are currently supported. The file names are defined in the list `syntherk.molfiles`, while the molecules are defined in `syntherk.molecule_ids`. Other molecules or different line lists can be included by editing these variables. For example, the Kurucz website also has a single large file that includes all diatomic molecules except TiO (diatomics.asc).

CH : `chmasseron.asc` (from Kurucz website, dated 2-Aug-2015)

MgH : `mgh15082018.asc` (from Kurucz website, dated 15-Aug-2015)

NH : `nhaxCas.asc`, `nhcaCas.asc` (the Kurucz website now has a single file, nh.asc, dated 11-Apr-2014)

OH : `ohupdate.asc` (from Kurucz website, dated 10-Apr-2014)

SiH : `sihaxsightly12012018.asc`, `sihxxsightly12012018.asc` (from Kurucz website, dated 12-Jan-2018)

H2 : `h2.asc` (dated 11-Apr-2014), `h2xx.asc` (from Kurucz website, dated 17-Jun-2015)

C2 : c2ax.asc, c2ba.asc, c2dabrookek.asc, c2ea.asc

CN : cnaxbrookek.asc, cnbxbrookek.asc, cnxx12brooke.asc

CO : coax.asc, coxx.asc

SiO : sioax.asc, sioex.asc, sioxx.asc

CaH : cah.asc

FeH : fehfx.asc

VO : voax.asc, vobx.asc, vocx.asc

TiO : `tioschwenke.bin`, `eschwenke.bin`: From Castelli website. The file with the energies, `eschwenke.bin`, can be produced from several ASCII files and a programme available at the Kurucz website.

H2O : `h2ofastfix.bin` (from Kurucz website, dated 09-feb-2012).

In addition to the line lists, the following files are needed (all directories given relative to `absetup.catpref`):

`cats/Castelli2015/molecules.dat`: List of data for molecules

`cats/Castelli2015/pfiron.dat`: Partition functions for iron-group elements (other partition functions are hard-coded)

`cats/Castelli2015/continua.dat`: Data for continuum opacity calculations

`cats/Castelli2015/linelists`: Predicted lines and other line lists for `ATLAS12`.

`cats/Castelli2015/dflines/*.bin`: Line lists for computation of ODFs. These can be generated from the files in `cats/Castelli2015/linelists` with the programmes `repack*.exe` (available at the Kurucz and Castelli websites)

`cats/Castelli2015/odfsl/*.bdf`: ODFs computed for Solar-scaled and alpha-enhanced composition in 0.25 dex steps of metallicity.

`cats/Castelli2015/atlas9/*odfnew.dat`: Pre-computed `ATLAS9` models for Solar-scaled and alpha-enhanced composition, computed by Castelli.

`cats/Kurucz/colors/*.dat`: Theoretical colours, from Castelli & Kurucz (2003). These are used when deriving stellar parameters from photometry.

3.2 Marcs atmospheres / TurboSpectrum

`ISPy3` uses the `interpol_modeles` programme (written by T. Masseron) to generate `MARCS` models for any specified combination of $T_{\rm eff}$, $\log g$, and [Fe/H] by interpolation in the pre-computed model library available via the `MARCS` website (`https://marcs.astro.uu.se/search.php`). `ISPy3` is configured to use the "standard composition" model grid, which has Solar-scaled composition at high metallicities and alpha-enhanced composition at lower metallicities. Spherical models are used for low surface gravities ($\log g < 3$) and plane-parallel models otherwise. The model grid covers temperatures in the range $2500\,{\rm K} \leq T_{\rm eff} \leq 8000\,{\rm K}$, surface gravities $-0.5 \leq \log g \leq 5.0$, and metallicities $-5.0 \leq {\rm [Fe/H]} \leq +0.75$. There are nearly 3000 spherical models with a microturbulent velocity of 2 km/s and a mass of 1 $M_\odot$, and 2400 plane-parallel models with a microturbulence of 1 km/s. The webform on the `MARCS` website will not allow all of these to be downloaded at once, so it is necessary to narrow down the search to a more limited set of parameters per request.

For temperatures in the range $3800\,{\rm K} \leq T_{\rm eff} \leq 7000\,{\rm K}$ an "optimised" (non-linear) interpolation scheme is used, while linear interpolation is used for temperatures outside this range. Details can be found in Masseron (2008). If a model outside the grid is requested, the closest point on the grid will be used - hence, it is not wise to request models with parameters that lie much outside those included in the grid. User discretion is advised!

For spherical geometry, the `MARCS` grid contains models for several different masses (0.5, 1, 2, 5, and 15 $M_\odot$); the physical radius of curvature then follows once the surface gravity is specified. `ISPy3` uses the models computed for a mass of 1 $M_\odot$ (for which the grid is most complete), and while this is probably appropriate for modelling of giants in old globular clusters, it may not be for cool supergiants in younger populations.

To compute synthetic spectra from the `MARCS` models, the `TurboSpectrum` spectral synthesis code (Alvaro & Plez 1998; Plez 2012) is used. The abundances in the model spectra are not restricted to those in the input atmosphere models. The latest (2019) version of TurboSpectrum is available at Github (`https://github.com/bertrandplez/Turbospectrum2019/releases`) while the 2015 version can be downloaded from the website of B. Plez (`https://www.lupm.in2p3.fr/users/plez/`) that also has extensive molecular line lists. The 2019 version of TurboSpectrum can use `ATLAS` model atmospheres, but is significantly slower than previous versions. I have mostly used and tested `ISPy3` with the 2014 version.

`Turbospectrum` uses a common file format for both atomic and molecular line lists, which is different than those used by the Kurucz codes. Atomic line lists in the Kurucz format can be converted to the `Turbospectrum` format with the `kur2bsyn.py` utility and molecular line lists in the Kurucz ASCII format can be converted with the `kmol2bsyn.py` utility. This way, consistency between `SYNTHE` and `Turbospectrum` can be achieved. Atomic line lists for `Turbospectrum` can also be downloaded from the VALD website (`http://vald.astro.uu.se/~vald/php/vald.php`) and put in the right format with utilities available at the Plez website. The TiO line list is a somewhat special case, as `SYNTHE` uses the Schwenke line list in binary format, whereas `Turbospectrum` instead uses the line list from the VALD database (Plez 1998, updated version from January 2012).

The `ISPy3` interface for `Turbospectrum` uses a similar sequence of initialisation and computation as `SYNTHE`, although the initialisation procedure is internally simpler for Turbospectrum. The equivalent of the `synbeg()` function is called `turbospec.turboinit()`, but does not involve execution of any external code and simply copies some files to the temporary directory. The synthetic spectra are then computed with a call to `turbospec.turbospec()`.

3.2.1 External executables for MARCS/TurboSpectrum

These must be available in the directory `absetup.binpath`.

babsma_lu Calculation of continuous absorption coefficients (2014 or 2015 version)

bsyn_lu Spectral synthesis code (2014 or 2015 version)

babsma_lu.19 Calculation of continuous absorption coefficients (2019 version)

bsyn_lu.19 Spectral synthesis code that can read `ATLAS` models (2019 version)

interpol_modeles Code to interpolate in `MARCS` model grid, available from the `MARCS` website (`https://marcs.astro.uu.se/software.php`). Compiled with the `optimize` option set to `.true.` (i.e. optimised interpolation for models in the range 3800 - 7000 K).

interpol_modeles_lin Same as above, but compiled with `optimize = .false.` (linear interpolation for models cooler than 3800 K).

3.2.2 Input files for MARCS/TurboSpectrum

All directories given relative to `absetup.catpref`.

`cats/Marcs/Standard_composition/Spherical/*mod.gz`: Spherical `MARCS` models, surface gravities $-0.5 \leq \log g \leq +3.5$, microturbulence of 2 km/s.

`cats/Marcs/Standard_composition/Plane-parallel/*mod.gz`: Plane-parallel `MARCS` models, surface gravities $+3.0 \leq \log g \leq +5.5$, microturbulence of 1 km/s.

In `turbospec.py`: `turbospec.atomdir` and `turbospec.moldir` point to directories containing atomic and molecular line lists in the BSYN format. `turbospec.plezdir` points to a directory containing various input files for `TurboSpectrum`, as well as the line lists for CH and TiO. The `DATA/` directory in the `Turbospectrum` distribution should be copied as a subdirectory under this directory.

4 Basic examples

This section provides some examples to illustrate how `ISPy3` can be used to compute stellar model atmospheres and synthetic spectra.

4.1 Computing an ATLAS9 model

To compute an ATLAS9 model for a red giant with an effective temperature of $T_{\rm eff} = 4200$ K, surface gravity $\log g = 2.0$, metallicity [m/H]=−0.5, and microturbulent velocity of 2 km/s, we use the following sequence of Python commands:

```
> from ispy3 import kurucz
> m, teff, logg, vturb = -0.5, 4200., 2.00, 2.0
> kurucz.mkodf(m, odfs='S')
> kurucz.mkatm(teff, logg, m, 't4200g200m050', vturb=vturb)
```

First, the kurucz module is imported and the stellar parameters defined. Next, kurucz.mkodf() is used to set up an ODF for the specified metallicity by interpolation in the pre-computed set of ODFs. The argument odfs='S' specifies that the Solar-scaled ODFs will be used; the other option is 'A' (alpha-enhanced ODFs). The ODFs are computed for a fixed set of microturbulent velocities: 0 km/s, 1 km/s, 2 km/s, 4 km/s, and 8 km/s, all of which are stored in the working directory (odfbig0.bdf, odfbig1.bdf, ..., odfbig8.bdf). Finally, kurucz.mkatm() selects the interpolated ODF with the closest matching microturbulence and computes the model atmosphere. Since we haven't specified otherwise, the initial guess for the structure of the model will be selected among the pre-existing Castelli models, which must therefore be available. The temperature structure of the pre-existing model will then be scaled to the temperature of the requested model, and iterations started from the rescaled initial guess. The final model will be stored in an output file with the name t4200g200m050.A9, where the extension is automatically added. There will also be a diagnostic output file called t4200g200m050.out. The computation time for this model (on a 2010 Mac Pro workstation) was about 9 s, with an additional 15 s to interpolate in the ODFs (that has to be done only once for a given metallicity).

To compute a model from scratch, we set kurucz.A9_FROMSCRATCH = True. The procedure is otherwise identical:

```
> from ispy3 import kurucz
> m, teff, logg, vturb = -0.5, 4200., 2.00, 2.0
> kurucz.mkodf(m)
> kurucz.A9_FROMSCRATCH = True
> kurucz.mkatm(teff, logg, m, 't4200g200m050s', vturb=vturb)
```

To compare the two models, we can inspect the .out files. Of interest here are the two last columns in the model output at the end of these files, which list the percentage error on the flux (per layer) and the flux derivative, both of which should be small if the model has converged. An important point to be aware of is that the ATLAS code does not itself test for convergence. It simply carries out the specified number of iterations, and it is up to the user to verify that the model has indeed reached a satisfactory degree of convergence.

ATLAS9 model with initial guess from Castelli library:

TEFF 4200. LOG G 2.00000 [-0.5] VTURB=2.0 L/H=1.25 NOVER NEW ODF ITERATION 15

0

	RHOX	TEMP	PRESSURE	ELECTRON NUMBER	DENSITY	ROSSELAND MEAN	HEIGHT (KM)	ROSSELAND DEPTH	FRACTION CONV FLUX	RADIATIVE ACCELERATION	PER CENT FLUX ERROR	PER CENT FLUX DERIV
1	8.183E-03	2362.8	8.183E-01	5.226E+06	5.222E-12	1.630E-05	0.000E+00	1.334E-07	0.000E+00	2.653E-03	0.004	2.905
2	1.087E-02	2388.7	1.087E+00	6.964E+06	6.862E-12	1.683E-05	4.528E+03	1.778E-07	0.000E+00	2.506E-03	0.004	4.353
3	1.426E-02	2412.0	1.426E+00	9.143E+06	8.917E-12	1.816E-05	8.900E+03	2.371E-07	0.000E+00	2.349E-03	0.004	4.765
4	1.846E-02	2435.4	1.846E+00	1.186E+07	1.144E-11	1.950E-05	1.309E+04	3.162E-07	0.000E+00	2.186E-03	0.004	3.079
5	2.372E-02	2459.8	2.372E+00	1.532E+07	1.455E-11	2.050E-05	1.716E+04	4.217E-07	0.000E+00	2.031E-03	0.004	2.027
6	3.048E-02	2485.7	3.048E+00	1.984E+07	1.850E-11	2.114E-05	2.127E+04	5.623E-07	0.000E+00	1.884E-03	0.004	1.210
7	3.914E-02	2512.1	3.915E+00	2.572E+07	2.351E-11	2.218E-05	2.542E+04	7.499E-07	0.000E+00	1.740E-03	0.004	0.204
8	4.999E-02	2538.6	4.999E+00	3.318E+07	2.972E-11	2.395E-05	2.951E+04	1.000E-06	0.000E+00	1.596E-03	0.004	-1.037
9	6.345E-02	2565.3	6.345E+00	4.262E+07	3.733E-11	2.552E-05	3.355E+04	1.334E-06	0.000E+00	1.472E-03	0.004	-2.883
10	8.039E-02	2592.5	8.040E+00	5.475E+07	4.680E-11	2.690E-05	3.759E+04	1.778E-06	0.000E+00	1.364E-03	0.004	-5.161
11	1.019E-01	2620.5	1.019E+01	7.050E+07	5.871E-11	2.818E-05	4.169E+04	2.371E-06	0.000E+00	1.271E-03	0.004	-7.835
12	1.288E-01	2649.3	1.288E+01	9.074E+07	7.338E-11	3.068E-05	4.578E+04	3.162E-06	0.000E+00	1.181E-03	0.004	-10.567
13	1.616E-01	2678.0	1.617E+01	1.162E+08	9.111E-11	3.346E-05	4.979E+04	4.217E-06	0.000E+00	1.103E-03	0.004	-10.530
14	2.019E-01	2706.5	2.020E+01	1.482E+08	1.126E-10	3.624E-05	5.376E+04	5.623E-06	0.000E+00	1.038E-03	0.004	-10.274
15	2.517E-01	2735.0	2.518E+01	1.888E+08	1.389E-10	3.903E-05	5.774E+04	7.499E-06	0.000E+00	9.833E-04	0.004	-11.367
16	3.132E-01	2764.6	3.133E+01	2.408E+08	1.711E-10	4.226E-05	6.172E+04	1.000E-05	0.000E+00	9.389E-04	0.004	-12.848
17	3.876E-01	2795.5	3.877E+01	3.071E+08	2.094E-10	4.734E-05	6.565E+04	1.334E-05	0.000E+00	9.009E-04	0.004	-10.547
18	4.764E-01	2826.0	4.765E+01	3.892E+08	2.545E-10	5.271E-05	6.949E+04	1.778E-05	0.000E+00	8.687E-04	0.004	-11.108
19	5.831E-01	2856.5	5.832E+01	4.917E+08	3.082E-10	5.841E-05	7.330E+04	2.371E-05	0.000E+00	8.420E-04	0.004	-11.748
20	7.114E-01	2887.9	7.116E+01	6.214E+08	3.718E-10	6.483E-05	7.708E+04	3.162E-05	0.000E+00	8.226E-04	0.004	-13.220
21	8.636E-01	2921.5	8.637E+01	7.872E+08	4.461E-10	7.380E-05	8.081E+04	4.217E-05	0.000E+00	8.128E-04	0.004	-10.598
22	1.042E+00	2954.8	1.042E+02	9.918E+08	5.323E-10	8.363E-05	8.447E+04	5.623E-05	0.000E+00	8.039E-04	0.004	-9.311
23	1.251E+00	2988.4	1.251E+02	1.246E+09	6.317E-10	9.586E-05	8.807E+04	7.499E-05	0.000E+00	8.007E-04	0.004	-7.748
24	1.494E+00	3022.7	1.494E+02	1.562E+09	7.457E-10	1.100E-04	9.161E+04	1.000E-04	0.000E+00	8.006E-04	0.003	-10.617
25	1.773E+00	3057.4	1.773E+02	1.949E+09	8.745E-10	1.296E-04	9.505E+04	1.334E-04	0.000E+00	8.122E-04	0.003	-7.380
26	2.089E+00	3090.8	2.089E+02	2.408E+09	1.019E-09	1.513E-04	9.840E+04	1.778E-04	0.000E+00	8.233E-04	0.003	-5.817
27	2.453E+00	3123.2	2.452E+02	2.957E+09	1.184E-09	1.754E-04	1.017E+05	2.371E-04	0.000E+00	8.358E-04	0.002	-4.151
28	2.871E+00	3154.8	2.871E+02	3.609E+09	1.372E-09	2.025E-04	1.050E+05	3.162E-04	0.000E+00	8.517E-04	0.002	-2.899
29	3.350E+00	3186.6	3.350E+02	4.392E+09	1.585E-09	2.388E-04	1.082E+05	4.217E-04	0.000E+00	8.798E-04	0.002	-1.898
30	3.890E+00	3217.0	3.889E+02	5.300E+09	1.823E-09	2.824E-04	1.114E+05	5.623E-04	0.000E+00	9.138E-04	0.002	-1.281
31	4.502E+00	3245.9	4.501E+02	6.343E+09	2.091E-09	3.306E-04	1.145E+05	7.499E-04	0.000E+00	9.504E-04	0.002	-0.825
32	5.202E+00	3273.5	5.201E+02	7.547E+09	2.395E-09	3.842E-04	1.177E+05	1.000E-03	0.000E+00	9.910E-04	0.001	-0.521
33	6.007E+00	3300.2	6.006E+02	8.941E+09	2.744E-09	4.441E-04	1.208E+05	1.334E-03	0.000E+00	1.037E-03	0.001	-0.342
34	6.935E+00	3326.1	6.934E+02	1.056E+10	3.143E-09	5.147E-04	1.240E+05	1.778E-03	0.000E+00	1.093E-03	0.001	-0.259
35	8.002E+00	3351.4	8.001E+02	1.243E+10	3.599E-09	5.971E-04	1.271E+05	2.371E-03	0.000E+00	1.160E-03	0.001	-0.182
36	9.231E+00	3375.8	9.230E+02	1.458E+10	4.123E-09	6.901E-04	1.303E+05	3.162E-03	0.000E+00	1.236E-03	0.001	-0.190
37	1.065E+01	3400.1	1.065E+03	1.707E+10	4.722E-09	7.981E-04	1.335E+05	4.217E-03	0.000E+00	1.326E-03	0.001	-0.195
38	1.228E+01	3423.8	1.228E+03	1.995E+10	5.410E-09	9.225E-04	1.368E+05	5.623E-03	0.000E+00	1.432E-03	0.000	-0.336
39	1.417E+01	3449.6	1.417E+03	2.342E+10	6.194E-09	1.069E-03	1.400E+05	7.499E-03	0.000E+00	1.555E-03	-0.002	-1.790
40	1.634E+01	3474.7	1.634E+03	2.742E+10	7.092E-09	1.235E-03	1.433E+05	1.000E-02	0.000E+00	1.691E-03	-0.007	-2.143
41	1.884E+01	3499.7	1.884E+03	3.205E+10	8.122E-09	1.430E-03	1.466E+05	1.334E-02	0.000E+00	1.851E-03	-0.012	-1.387
42	2.173E+01	3526.3	2.172E+03	3.755E+10	9.295E-09	1.658E-03	1.499E+05	1.778E-02	0.000E+00	2.040E-03	-0.016	-0.495
43	2.503E+01	3555.5	2.503E+03	4.423E+10	1.062E-08	1.931E-03	1.532E+05	2.371E-02	0.000E+00	2.266E-03	-0.017	0.167
44	2.881E+01	3588.5	2.880E+03	5.245E+10	1.211E-08	2.263E-03	1.565E+05	3.162E-02	0.000E+00	2.537E-03	-0.015	0.347
45	3.308E+01	3627.0	3.308E+03	6.278E+10	1.376E-08	2.677E-03	1.598E+05	4.217E-02	0.000E+00	2.864E-03	-0.013	0.030
46	3.788E+01	3670.1	3.788E+03	7.557E+10	1.557E-08	3.192E-03	1.631E+05	5.623E-02	0.000E+00	3.252E-03	-0.012	0.036
47	4.323E+01	3719.7	4.323E+03	9.171E+10	1.753E-08	3.835E-03	1.664E+05	7.499E-02	0.000E+00	3.718E-03	-0.011	0.017
48	4.915E+01	3776.6	4.914E+03	1.121E+11	1.962E-08	4.634E-03	1.695E+05	1.000E-01	0.000E+00	4.276E-03	-0.011	0.011
49	5.566E+01	3842.5	5.566E+03	1.381E+11	2.183E-08	5.625E-03	1.727E+05	1.334E-01	0.000E+00	4.925E-03	-0.010	0.004
50	6.281E+01	3918.2	6.280E+03	1.710E+11	2.414E-08	6.845E-03	1.758E+05	1.778E-01	0.000E+00	5.681E-03	-0.009	0.000
51	7.064E+01	4005.1	7.063E+03	2.122E+11	2.655E-08	8.317E-03	1.789E+05	2.371E-01	0.000E+00	6.545E-03	-0.008	-0.003
52	7.927E+01	4104.6	7.926E+03	2.631E+11	2.906E-08	1.001E-02	1.820E+05	3.162E-01	0.000E+00	7.460E-03	-0.006	-0.006
53	8.889E+01	4217.9	8.888E+03	3.250E+11	3.169E-08	1.191E-02	1.852E+05	4.217E-01	0.000E+00	8.424E-03	-0.006	-0.026
54	9.976E+01	4346.7	9.975E+03	3.992E+11	3.450E-08	1.394E-02	1.885E+05	5.623E-01	0.000E+00	9.389E-03	-0.001	-0.005
55	1.123E+02	4492.6	1.123E+04	4.888E+11	3.756E-08	1.602E-02	1.919E+05	7.499E-01	0.000E+00	1.031E-02	0.002	-0.053
56	1.268E+02	4658.1	1.268E+04	6.055E+11	4.090E-08	1.845E-02	1.956E+05	1.000E+00	0.000E+00	1.143E-02	-0.000	-0.093
57	1.433E+02	4846.7	1.433E+04	7.931E+11	4.442E-08	2.222E-02	1.995E+05	1.334E+00	1.086E-08	1.335E-02	-0.003	-0.165
58	1.607E+02	5060.7	1.607E+04	1.175E+12	4.769E-08	2.988E-02	2.033E+05	1.778E+00	5.072E-06	1.755E-02	0.002	-0.107
59	1.768E+02	5305.5	1.767E+04	2.028E+12	5.004E-08	4.610E-02	2.066E+05	2.371E+00	3.078E-04	2.672E-02	-0.008	0.000
60	1.899E+02	5585.5	1.899E+04	3.953E+12	5.104E-08	7.918E-02	2.091E+05	3.162E+00	3.414E-03	4.565E-02	0.013	-0.001
61	1.997E+02	5905.8	1.996E+04	8.302E+12	5.076E-08	1.455E-01	2.111E+05	4.217E+00	1.738E-02	8.309E-02	-0.032	0.000
62	2.066E+02	6259.1	2.066E+04	1.769E+13	4.954E-08	2.749E-01	2.125E+05	5.623E+00	8.628E-02	1.467E-01	0.048	-0.000
63	2.117E+02	6606.0	2.116E+04	3.473E+13	4.804E-08	4.946E-01	2.135E+05	7.499E+00	2.613E-01	2.135E-01	-0.068	-0.000
64	2.156E+02	6895.4	2.156E+04	5.813E+13	4.683E-08	7.909E-01	2.143E+05	1.000E+01	4.847E-01	2.387E-01	0.065	0.000
65	2.191E+02	7129.0	2.190E+04	8.566E+13	4.596E-08	1.142E+00	2.151E+05	1.334E+01	6.554E-01	2.307E-01	-0.040	-0.001
66	2.224E+02	7321.6	2.223E+04	1.159E+14	4.537E-08	1.538E+00	2.158E+05	1.778E+01	7.578E-01	2.190E-01	-0.006	0.008
67	2.258E+02	7497.0	2.257E+04	1.508E+14	4.490E-08	2.008E+00	2.165E+05	2.371E+01	8.236E-01	2.085E-01	0.048	-0.000
68	2.292E+02	7654.2	2.292E+04	1.892E+14	4.457E-08	2.543E+00	2.173E+05	3.162E+01	8.687E-01	1.963E-01	-0.044	-0.000
69	2.329E+02	7806.9	2.328E+04	2.338E+14	4.431E-08	3.192E+00	2.181E+05	4.217E+01	9.009E-01	1.862E-01	0.012	0.001
70	2.369E+02	7948.3	2.368E+04	2.828E+14	4.416E-08	3.932E+00	2.190E+05	5.623E+01	9.240E-01	1.760E-01	0.009	-0.000
71	2.412E+02	8092.5	2.411E+04	3.411E+14	4.403E-08	4.851E+00	2.200E+05	7.499E+01	9.416E-01	1.667E-01	0.003	-0.000
72	2.458E+02	8224.6	2.457E+04	4.031E+14	4.403E-08	5.869E+00	2.211E+05	1.000E+02	9.508E-01	1.696E-01	-0.019	-0.001

ATLAS9 model computed from scratch:

TEFF 4200. LOG G 2.00000 [-0.5] VTURB=2.0 L/H=1.25 NOVER NEW ODF ITERATION 15

0

	RHOX	TEMP	PRESSURE	ELECTRON NUMBER	DENSITY	ROSSELAND MEAN	HEIGHT (KM)	ROSSELAND DEPTH	FRACTION CONV FLUX	RADIATIVE ACCELERATION	PER CENT FLUX ERROR	PER CENT FLUX DERIV
1	9.894E-03	2411.4	1.082E+00	7.708E+06	6.915E-12	1.236E-05	0.000E+00	1.334E-07	0.000E+00	2.034E-03	0.009	*********
2	1.333E-02	2474.2	1.449E+00	1.159E+07	8.987E-12	1.198E-05	4.855E+03	1.778E-07	0.000E+00	1.761E-03	0.008	*********
3	1.778E-02	2487.3	1.919E+00	1.472E+07	1.185E-11	1.329E-05	9.530E+03	2.371E-07	0.000E+00	1.686E-03	0.007	*********
4	2.310E-02	2500.2	2.478E+00	1.810E+07	1.521E-11	1.516E-05	1.377E+04	3.162E-07	0.000E+00	1.669E-03	0.006	-6730.372
5	2.960E-02	2524.3	3.144E+00	2.281E+07	1.907E-11	1.646E-05	1.769E+04	4.217E-07	0.000E+00	1.606E-03	0.006	-4606.210
6	3.768E-02	2549.7	3.962E+00	2.889E+07	2.373E-11	1.787E-05	2.153E+04	5.623E-07	0.000E+00	1.531E-03	0.005	-3376.736
7	4.778E-02	2581.6	4.985E+00	3.744E+07	2.941E-11	1.868E-05	2.542E+04	7.499E-07	0.000E+00	1.431E-03	0.005	-3350.106
8	6.082E-02	2618.9	6.304E+00	4.959E+07	3.655E-11	1.926E-05	2.945E+04	1.000E-06	0.000E+00	1.327E-03	0.004	-3561.478
9	7.743E-02	2656.0	7.979E+00	6.570E+07	4.552E-11	2.059E-05	3.356E+04	1.334E-06	0.000E+00	1.236E-03	0.003	-3328.005
10	9.840E-02	2696.4	1.008E+01	8.804E+07	5.650E-11	2.179E-05	3.769E+04	1.778E-06	0.000E+00	1.142E-03	0.001	-3354.180
11	1.245E-01	2731.0	1.269E+01	1.160E+08	7.016E-11	2.374E-05	4.185E+04	2.371E-06	0.000E+00	1.052E-03	-0.001	-2988.694
12	1.560E-01	2759.0	1.583E+01	1.484E+08	8.655E-11	2.654E-05	4.587E+04	3.162E-06	0.000E+00	9.889E-04	-0.003	-2352.597
13	1.937E-01	2787.4	1.958E+01	1.889E+08	1.058E-10	2.973E-05	4.978E+04	4.217E-06	0.000E+00	9.370E-04	-0.005	-1932.162
14	2.386E-01	2813.0	2.405E+01	2.373E+08	1.287E-10	3.303E-05	5.361E+04	5.623E-06	0.000E+00	8.866E-04	-0.008	-1586.224
15	2.927E-01	2835.2	2.945E+01	2.934E+08	1.564E-10	3.651E-05	5.742E+04	7.499E-06	0.000E+00	8.437E-04	-0.010	-1241.031
16	3.576E-01	2855.7	3.592E+01	3.589E+08	1.895E-10	4.081E-05	6.119E+04	1.000E-05	0.000E+00	8.156E-04	-0.013	-913.117
17	4.347E-01	2877.1	4.361E+01	4.380E+08	2.283E-10	4.600E-05	6.488E+04	1.334E-05	0.000E+00	8.015E-04	-0.016	-667.850
18	5.253E-01	2901.7	5.261E+01	5.370E+08	2.731E-10	5.274E-05	6.848E+04	1.778E-05	0.000E+00	7.947E-04	-0.018	-499.952
19	6.310E-01	2925.9	6.312E+01	6.551E+08	3.248E-10	6.002E-05	7.199E+04	2.371E-05	0.000E+00	7.863E-04	-0.021	-377.925
20	7.550E-01	2950.3	7.548E+01	7.975E+08	3.852E-10	6.789E-05	7.548E+04	3.162E-05	0.000E+00	7.792E-04	-0.023	-283.060
21	9.017E-01	2974.8	9.009E+01	9.693E+08	4.561E-10	7.646E-05	7.896E+04	4.217E-05	0.000E+00	7.759E-04	-0.026	-205.891
22	1.075E+00	3001.2	1.073E+02	1.183E+09	5.388E-10	8.660E-05	8.244E+04	5.623E-05	0.000E+00	7.776E-04	-0.028	-152.356
23	1.277E+00	3029.7	1.274E+02	1.450E+09	6.336E-10	1.004E-04	8.588E+04	7.499E-05	0.000E+00	7.927E-04	-0.031	-116.764
24	1.506E+00	3065.8	1.502E+02	1.811E+09	7.380E-10	1.188E-04	8.921E+04	1.000E-04	0.000E+00	7.974E-04	-0.034	-122.721
25	1.768E+00	3090.5	1.763E+02	2.171E+09	8.593E-10	1.368E-04	9.248E+04	1.334E-04	0.000E+00	7.951E-04	-0.037	-87.612
26	2.072E+00	3116.8	2.066E+02	2.608E+09	9.984E-10	1.572E-04	9.574E+04	1.778E-04	0.000E+00	8.054E-04	-0.041	-60.832
27	2.425E+00	3142.8	2.417E+02	3.126E+09	1.159E-09	1.802E-04	9.901E+04	2.371E-04	0.000E+00	8.176E-04	-0.043	-40.720
28	2.834E+00	3169.2	2.825E+02	3.746E+09	1.343E-09	2.075E-04	1.023E+05	3.162E-04	0.000E+00	8.362E-04	-0.046	-25.405
29	3.304E+00	3196.1	3.295E+02	4.482E+09	1.554E-09	2.424E-04	1.055E+05	4.217E-04	0.000E+00	8.654E-04	-0.048	-14.323
30	3.838E+00	3223.4	3.829E+02	5.355E+09	1.790E-09	2.843E-04	1.087E+05	5.623E-04	0.000E+00	9.004E-04	-0.050	-8.624
31	4.448E+00	3249.9	4.438E+02	6.365E+09	2.058E-09	3.311E-04	1.119E+05	7.499E-04	0.000E+00	9.377E-04	-0.051	-4.924
32	5.147E+00	3275.8	5.138E+02	7.542E+09	2.364E-09	3.837E-04	1.151E+05	1.000E-03	0.000E+00	9.796E-04	-0.052	-2.700
33	5.954E+00	3301.4	5.945E+02	8.916E+09	2.715E-09	4.428E-04	1.182E+05	1.334E-03	0.000E+00	1.027E-03	-0.052	-1.551
34	6.885E+00	3326.6	6.876E+02	1.052E+10	3.116E-09	5.129E-04	1.214E+05	1.778E-03	0.000E+00	1.084E-03	-0.053	-0.881
35	7.955E+00	3351.3	7.947E+02	1.237E+10	3.575E-09	5.946E-04	1.246E+05	2.371E-03	0.000E+00	1.151E-03	-0.053	-0.435
36	9.189E+00	3375.3	9.181E+02	1.451E+10	4.102E-09	6.871E-04	1.279E+05	3.162E-03	0.000E+00	1.228E-03	-0.053	-0.272
37	1.061E+01	3399.3	1.060E+03	1.699E+10	4.705E-09	7.946E-04	1.311E+05	4.217E-03	0.000E+00	1.318E-03	-0.054	-0.135
38	1.225E+01	3422.7	1.225E+03	1.986E+10	5.397E-09	9.186E-04	1.344E+05	5.623E-03	0.000E+00	1.425E-03	-0.054	-0.279
39	1.414E+01	3449.3	1.414E+03	2.338E+10	6.181E-09	1.067E-03	1.376E+05	7.499E-03	0.000E+00	1.547E-03	-0.057	-2.201
40	1.632E+01	3472.3	1.632E+03	2.722E+10	7.089E-09	1.226E-03	1.409E+05	1.000E-02	0.000E+00	1.676E-03	-0.061	-1.250
41	1.885E+01	3496.1	1.884E+03	3.175E+10	8.132E-09	1.417E-03	1.442E+05	1.334E-02	0.000E+00	1.835E-03	-0.062	0.595
42	2.176E+01	3521.9	2.175E+03	3.716E+10	9.321E-09	1.641E-03	1.476E+05	1.778E-02	0.000E+00	2.026E-03	-0.055	2.051
43	2.509E+01	3551.4	2.509E+03	4.383E+10	1.066E-08	1.914E-03	1.509E+05	2.371E-02	0.000E+00	2.258E-03	-0.041	2.517
44	2.889E+01	3586.6	2.889E+03	5.228E+10	1.216E-08	2.256E-03	1.543E+05	3.162E-02	0.000E+00	2.539E-03	-0.027	1.411
45	3.317E+01	3626.8	3.317E+03	6.286E+10	1.380E-08	2.680E-03	1.576E+05	4.217E-02	0.000E+00	2.870E-03	-0.020	-0.012
46	3.796E+01	3669.5	3.796E+03	7.558E+10	1.561E-08	3.193E-03	1.608E+05	5.623E-02	0.000E+00	3.253E-03	-0.020	0.107
47	4.331E+01	3719.5	4.331E+03	9.178E+10	1.756E-08	3.838E-03	1.641E+05	7.499E-02	0.000E+00	3.721E-03	-0.018	0.036
48	4.922E+01	3776.3	4.922E+03	1.122E+11	1.965E-08	4.636E-03	1.672E+05	1.000E-01	0.000E+00	4.278E-03	-0.017	0.035
49	5.573E+01	3842.3	5.573E+03	1.382E+11	2.186E-08	5.628E-03	1.704E+05	1.334E-01	0.000E+00	4.928E-03	-0.016	0.016
50	6.287E+01	3918.0	6.287E+03	1.711E+11	2.417E-08	6.848E-03	1.735E+05	1.778E-01	0.000E+00	5.684E-03	-0.014	0.009
51	7.070E+01	4005.0	7.070E+03	2.123E+11	2.657E-08	8.321E-03	1.766E+05	2.371E-01	0.000E+00	6.548E-03	-0.013	0.003
52	7.933E+01	4104.5	7.933E+03	2.632E+11	2.908E-08	1.002E-02	1.797E+05	3.162E-01	0.000E+00	7.464E-03	-0.011	-0.000
53	8.894E+01	4217.8	8.894E+03	3.251E+11	3.172E-08	1.192E-02	1.828E+05	4.217E-01	0.000E+00	8.427E-03	-0.009	-0.017
54	9.981E+01	4346.6	9.981E+03	3.993E+11	3.452E-08	1.395E-02	1.861E+05	5.623E-01	0.000E+00	9.393E-03	-0.003	-0.002
55	1.123E+02	4492.6	1.123E+04	4.890E+11	3.758E-08	1.603E-02	1.896E+05	7.499E-01	0.000E+00	1.031E-02	-0.004	-0.088
56	1.268E+02	4658.0	1.268E+04	6.056E+11	4.091E-08	1.845E-02	1.933E+05	1.000E+00	0.000E+00	1.143E-02	-0.011	-0.078
57	1.433E+02	4846.4	1.433E+04	7.930E+11	4.443E-08	2.222E-02	1.971E+05	1.334E+00	1.096E-08	1.335E-02	0.014	-0.060
58	1.607E+02	5061.2	1.607E+04	1.176E+12	4.770E-08	2.990E-02	2.009E+05	1.778E+00	5.066E-06	1.756E-02	0.001	-0.265
59	1.768E+02	5305.1	1.768E+04	2.026E+12	5.005E-08	4.608E-02	2.042E+05	2.371E+00	3.088E-04	2.670E-02	-0.029	0.000
60	1.899E+02	5585.7	1.899E+04	3.955E+12	5.105E-08	7.921E-02	2.068E+05	3.162E+00	3.419E-03	4.568E-02	0.019	0.001
61	1.997E+02	5905.4	1.997E+04	8.296E+12	5.077E-08	1.455E-01	2.087E+05	4.217E+00	1.738E-02	8.307E-02	-0.011	-0.001
62	2.067E+02	6259.6	2.066E+04	1.771E+13	4.954E-08	2.751E-01	2.101E+05	5.623E+00	8.607E-02	1.468E-01	-0.000	-0.004
63	2.117E+02	6605.3	2.117E+04	3.468E+13	4.805E-08	4.941E-01	2.111E+05	7.499E+00	2.611E-01	2.133E-01	-0.068	0.005
64	2.156E+02	6896.1	2.156E+04	5.820E+13	4.683E-08	7.919E-01	2.120E+05	1.000E+01	4.840E-01	2.396E-01	0.189	-0.002
65	2.191E+02	7129.8	2.191E+04	8.575E+13	4.596E-08	1.143E+00	2.127E+05	1.334E+01	6.543E-01	2.310E-01	-0.325	-0.001
66	2.224E+02	7322.4	2.224E+04	1.161E+14	4.537E-08	1.539E+00	2.134E+05	1.778E+01	7.593E-01	2.183E-01	0.139	0.054
67	2.258E+02	7496.4	2.257E+04	1.507E+14	4.491E-08	2.007E+00	2.142E+05	2.371E+01	8.212E-01	2.116E-01	0.347	-0.021
68	2.293E+02	7659.9	2.292E+04	1.906E+14	4.454E-08	2.564E+00	2.150E+05	3.162E+01	8.702E-01	1.932E-01	-1.301	0.001
69	2.329E+02	7798.1	2.329E+04	2.311E+14	4.437E-08	3.151E+00	2.158E+05	4.217E+01	9.007E-01	1.866E-01	1.304	0.004
70	2.369E+02	7961.2	2.368E+04	2.875E+14	4.407E-08	4.007E+00	2.167E+05	5.623E+01	9.224E-01	1.830E-01	0.123	-0.097
71	2.411E+02	8087.2	2.410E+04	3.387E+14	4.406E-08	4.812E+00	2.176E+05	7.499E+01	9.436E-01	1.566E-01	-2.040	0.003
72	2.458E+02	8225.3	2.457E+04	4.034E+14	4.403E-08	5.875E+00	2.187E+05	1.000E+02	9.501E-01	1.778E-01	3.026	0.203

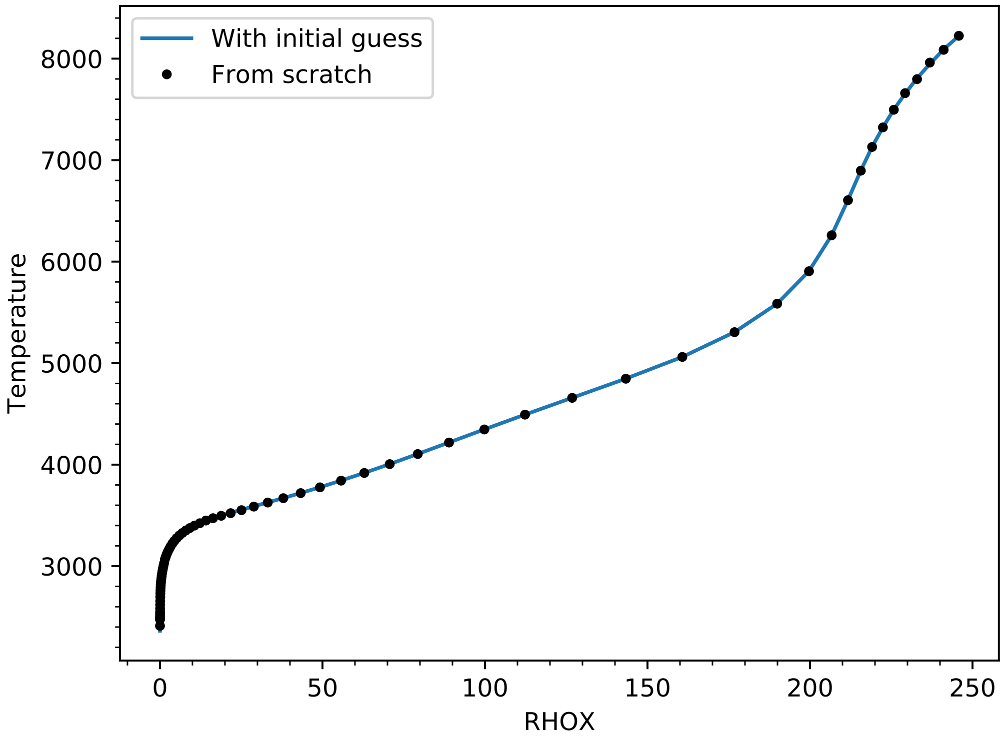

Figure 1: Comparison of two ATLAS9 models for identical physical parameters ($T_{\rm eff}$ = 4200 K, $\log g$ = 2.0, [m/H]= −0.5, but different initial conditions (initial guess vs. model computed from scratch).

The errors on the fluxes are small in both cases, but the flux derivative hasn't converged quite as well for the model that was computed from scratch, especially in the outermost layers. This is because a model calculation from scratch starts from a grey atmosphere, which will in general be a worse starting guess than a scaled pre-existing ATLAS model with parameters close to those of the requested model. Hence, more iterations are necessary. To achieve better convergence, one can increase the number of iterations to `kurucz.A9_NREP = 3`. Nevertheless, the structure of the two models is already quite similar, as can be seen when plotting the temperature as a function of depth (RHOX) in Figure 1.

4.2 Computing an ATLAS12 model

The procedure for computing an ATLAS12 model is similar to that for ATLAS9, except that the step of setting up the ODF can be skipped:

```
> from ispy3 import kurucz
```

```
> m, teff, logg, vturb = -0.5, 4200., 2.00, 2.0
> kurucz.mkatm_a12(teff, logg, m, 't4200g200m050', vturb=vturb)
```

A major difference is the time required to compute the model, which in this example was 50 *minutes* (as compared with 9 *seconds* for the equivalent `ATLAS9` model). The computation time will vary strongly depending on the temperature and composition of the models, with cooler and more metal-rich models generally taking longer to compute as more lines contribute significantly to the opacity. The model atmosphere is now stored in `t4200g200m050.A12` and the diagnostic output file is `t4200g200m050b.out`. The last few lines in this file are similar to those produced for the `ATLAS9` models, as can be seen below. The model has converged nicely, and its structure is very similar to that of the `ATLAS9` model, as expected. In the outermost and in the very deepest layers, the temperature difference between the `ATLAS9` and `ATLAS12` models reaches 15 K, but over most of the depth range it is less than 5 K.

ATLAS12 model:

TEFF 4200. LOG G 2.00000 ATLAS12 ITERATION 15

0

	RHOX	TEMP	PRESSURE	ELECTRON NUMBER	DENSITY	ROSSELAND MEAN	HEIGHT (KM)	ROSSELAND DEPTH	FRACTION CONV FLUX	RADIATIVE ACCELERATION	PER CENT FLUX ERROR	DERIV
1	8.528E-03	2364.5	8.527E-01	5.431E+06	5.439E-12	1.564E-05	0.000E+00	1.334E-07	0.000E+00	3.264E-03	-0.000	-5.562
2	1.129E-02	2389.6	1.129E+00	7.208E+06	7.128E-12	1.653E-05	4.483E+03	1.778E-07	0.000E+00	3.038E-03	-0.000	-2.182
3	1.478E-02	2414.0	1.478E+00	9.459E+06	9.236E-12	1.748E-05	8.817E+03	2.371E-07	0.000E+00	2.819E-03	-0.000	-2.154
4	1.919E-02	2438.9	1.919E+00	1.234E+07	1.187E-11	1.842E-05	1.306E+04	3.162E-07	0.000E+00	2.595E-03	-0.000	-2.980
5	2.477E-02	2464.5	2.476E+00	1.603E+07	1.516E-11	1.936E-05	1.721E+04	4.217E-07	0.000E+00	2.377E-03	-0.000	-2.472
6	3.185E-02	2490.8	3.184E+00	2.078E+07	1.929E-11	2.034E-05	2.134E+04	5.623E-07	0.000E+00	2.168E-03	-0.000	-2.189
7	4.083E-02	2517.7	4.083E+00	2.692E+07	2.447E-11	2.140E-05	2.546E+04	7.499E-07	0.000E+00	1.975E-03	-0.000	-2.447
8	5.219E-02	2545.2	5.219E+00	3.483E+07	3.094E-11	2.258E-05	2.959E+04	1.000E-06	0.000E+00	1.801E-03	-0.000	-2.403
9	6.653E-02	2573.1	6.654E+00	4.503E+07	3.902E-11	2.390E-05	3.371E+04	1.334E-06	0.000E+00	1.646E-03	-0.000	-3.478
10	8.457E-02	2601.5	8.457E+00	5.814E+07	4.906E-11	2.541E-05	3.782E+04	1.778E-06	0.000E+00	1.509E-03	-0.000	-4.014
11	1.071E-01	2630.2	1.071E+01	7.497E+07	6.147E-11	2.714E-05	4.192E+04	2.371E-06	0.000E+00	1.390E-03	-0.000	-3.679
12	1.352E-01	2659.3	1.352E+01	9.650E+07	7.674E-11	2.915E-05	4.600E+04	3.162E-06	0.000E+00	1.288E-03	-0.000	-3.230
13	1.700E-01	2688.7	1.700E+01	1.240E+08	9.543E-11	3.147E-05	5.006E+04	4.217E-06	0.000E+00	1.200E-03	-0.000	-2.292
14	2.129E-01	2718.4	2.129E+01	1.590E+08	1.182E-10	3.416E-05	5.409E+04	5.623E-06	0.000E+00	1.124E-03	-0.000	-2.532
15	2.653E-01	2748.5	2.653E+01	2.034E+08	1.457E-10	3.728E-05	5.808E+04	7.499E-06	0.000E+00	1.061E-03	-0.000	-2.326
16	3.293E-01	2778.9	3.293E+01	2.596E+08	1.788E-10	4.093E-05	6.204E+04	1.000E-05	0.000E+00	1.007E-03	-0.000	-2.461
17	4.067E-01	2809.9	4.067E+01	3.308E+08	2.184E-10	4.519E-05	6.595E+04	1.334E-05	0.000E+00	9.628E-04	-0.000	-2.061
18	5.000E-01	2841.4	4.999E+01	4.207E+08	2.655E-10	5.021E-05	6.982E+04	1.778E-05	0.000E+00	9.261E-04	-0.000	-2.043
19	6.115E-01	2873.5	6.114E+01	5.340E+08	3.211E-10	5.616E-05	7.363E+04	2.371E-05	0.000E+00	8.963E-04	-0.000	-1.831
20	7.439E-01	2906.2	7.439E+01	6.761E+08	3.862E-10	6.327E-05	7.739E+04	3.162E-05	0.000E+00	8.733E-04	-0.000	-2.030
21	9.001E-01	2939.3	9.001E+01	8.538E+08	4.620E-10	7.182E-05	8.108E+04	4.217E-05	0.000E+00	8.564E-04	-0.000	-1.813
22	1.083E+00	2972.8	1.083E+02	1.074E+09	5.495E-10	8.214E-05	8.470E+04	5.623E-05	0.000E+00	8.449E-04	-0.000	-1.818
23	1.295E+00	3006.2	1.295E+02	1.346E+09	6.498E-10	9.461E-05	8.825E+04	7.499E-05	0.000E+00	8.393E-04	-0.000	-1.610
24	1.540E+00	3039.8	1.540E+02	1.678E+09	7.641E-10	1.097E-04	9.172E+04	1.000E-04	0.000E+00	8.384E-04	-0.000	-2.170
25	1.821E+00	3072.7	1.821E+02	2.078E+09	8.938E-10	1.277E-04	9.512E+04	1.334E-04	0.000E+00	8.421E-04	-0.000	-1.933
26	2.143E+00	3105.0	2.143E+02	2.556E+09	1.040E-09	1.492E-04	9.845E+04	1.778E-04	0.000E+00	8.507E-04	-0.001	-1.671
27	2.509E+00	3136.5	2.509E+02	3.123E+09	1.206E-09	1.746E-04	1.017E+05	2.371E-04	0.000E+00	8.642E-04	-0.001	-1.294
28	2.927E+00	3167.1	2.927E+02	3.791E+09	1.393E-09	2.044E-04	1.049E+05	3.162E-04	0.000E+00	8.828E-04	-0.001	-0.892
29	3.403E+00	3196.8	3.403E+02	4.573E+09	1.604E-09	2.391E-04	1.081E+05	4.217E-04	0.000E+00	9.063E-04	-0.001	-0.609
30	3.945E+00	3225.5	3.945E+02	5.483E+09	1.844E-09	2.794E-04	1.113E+05	5.623E-04	0.000E+00	9.352E-04	-0.001	-0.399
31	4.565E+00	3253.4	4.565E+02	6.540E+09	2.115E-09	3.260E-04	1.144E+05	7.499E-04	0.000E+00	9.698E-04	-0.001	-0.268
32	5.274E+00	3280.3	5.274E+02	7.762E+09	2.424E-09	3.795E-04	1.175E+05	1.000E-03	0.000E+00	1.011E-03	-0.001	-0.196
33	6.087E+00	3306.3	6.087E+02	9.176E+09	2.775E-09	4.410E-04	1.207E+05	1.334E-03	0.000E+00	1.059E-03	-0.001	-0.141
34	7.021E+00	3331.6	7.021E+02	1.081E+10	3.177E-09	5.114E-04	1.238E+05	1.778E-03	0.000E+00	1.115E-03	-0.001	-0.125
35	8.096E+00	3356.3	8.096E+02	1.269E+10	3.637E-09	5.921E-04	1.270E+05	2.371E-03	0.000E+00	1.180E-03	-0.001	-0.089
36	9.335E+00	3380.5	9.334E+02	1.487E+10	4.163E-09	6.848E-04	1.301E+05	3.162E-03	0.000E+00	1.256E-03	-0.001	-0.094
37	1.076E+01	3404.5	1.076E+03	1.739E+10	4.767E-09	7.916E-04	1.334E+05	4.217E-03	0.000E+00	1.343E-03	-0.001	-0.089
38	1.241E+01	3428.3	1.241E+03	2.032E+10	5.460E-09	9.143E-04	1.366E+05	5.623E-03	0.000E+00	1.445E-03	-0.001	-0.118
39	1.431E+01	3453.0	1.431E+03	2.378E+10	6.252E-09	1.058E-03	1.398E+05	7.499E-03	0.000E+00	1.565E-03	-0.002	-0.566
40	1.651E+01	3477.7	1.651E+03	2.780E+10	7.160E-09	1.224E-03	1.431E+05	1.000E-02	0.000E+00	1.701E-03	-0.003	-0.476
41	1.903E+01	3503.2	1.903E+03	3.253E+10	8.196E-09	1.418E-03	1.464E+05	1.334E-02	0.000E+00	1.861E-03	-0.004	-0.194
42	2.194E+01	3530.4	2.194E+03	3.817E+10	9.375E-09	1.647E-03	1.497E+05	1.778E-02	0.000E+00	2.051E-03	-0.004	0.057
43	2.526E+01	3560.2	2.526E+03	4.500E+10	1.071E-08	1.923E-03	1.530E+05	2.371E-02	0.000E+00	2.278E-03	-0.004	0.110
44	2.905E+01	3593.6	2.905E+03	5.337E+10	1.220E-08	2.259E-03	1.563E+05	3.162E-02	0.000E+00	2.547E-03	-0.003	0.011
45	3.333E+01	3631.0	3.333E+03	6.368E+10	1.385E-08	2.669E-03	1.596E+05	4.217E-02	0.000E+00	2.868E-03	-0.003	-0.032
46	3.815E+01	3673.7	3.815E+03	7.652E+10	1.566E-08	3.177E-03	1.629E+05	5.623E-02	0.000E+00	3.253E-03	-0.003	-0.008
47	4.353E+01	3722.8	4.353E+03	9.274E+10	1.763E-08	3.812E-03	1.661E+05	7.499E-02	0.000E+00	3.714E-03	-0.003	-0.023
48	4.948E+01	3779.4	4.948E+03	1.133E+11	1.974E-08	4.604E-03	1.693E+05	1.000E-01	0.000E+00	4.263E-03	-0.003	-0.016
49	5.604E+01	3845.0	5.604E+03	1.394E+11	2.196E-08	5.593E-03	1.725E+05	1.334E-01	0.000E+00	4.914E-03	-0.003	-0.021
50	6.322E+01	3920.5	6.322E+03	1.724E+11	2.429E-08	6.808E-03	1.756E+05	1.778E-01	0.000E+00	5.668E-03	-0.002	-0.014
51	7.110E+01	4007.4	7.109E+03	2.139E+11	2.671E-08	8.271E-03	1.787E+05	2.371E-01	0.000E+00	6.520E-03	-0.001	-0.025
52	7.977E+01	4106.9	7.977E+03	2.651E+11	2.922E-08	9.968E-03	1.818E+05	3.162E-01	0.000E+00	7.442E-03	0.001	-0.037
53	8.944E+01	4220.5	8.943E+03	3.274E+11	3.187E-08	1.186E-02	1.849E+05	4.217E-01	0.000E+00	8.397E-03	0.004	-0.050
54	1.004E+02	4349.3	1.004E+04	4.020E+11	3.469E-08	1.387E-02	1.882E+05	5.623E-01	0.000E+00	9.344E-03	0.002	-0.106
55	1.129E+02	4495.1	1.129E+04	4.920E+11	3.776E-08	1.598E-02	1.917E+05	7.499E-01	0.000E+00	1.029E-02	-0.000	-0.088
56	1.275E+02	4660.7	1.275E+04	6.094E+11	4.109E-08	1.842E-02	1.954E+05	1.000E+00	0.000E+00	1.142E-02	0.009	-0.081
57	1.440E+02	4849.2	1.440E+04	7.984E+11	4.461E-08	2.222E-02	1.992E+05	1.334E+00	1.292E-08	1.334E-02	0.011	-0.168
58	1.614E+02	5063.3	1.613E+04	1.183E+12	4.787E-08	2.994E-02	2.030E+05	1.778E+00	5.575E-06	1.758E-02	-0.003	-0.135
59	1.774E+02	5307.8	1.774E+04	2.042E+12	5.019E-08	4.623E-02	2.063E+05	2.371E+00	3.137E-04	2.681E-02	0.007	0.012
60	1.905E+02	5588.6	1.904E+04	3.988E+12	5.117E-08	7.954E-02	2.088E+05	3.162E+00	3.118E-03	4.590E-02	0.014	-0.002
61	2.002E+02	5907.7	2.002E+04	8.351E+12	5.088E-08	1.460E-01	2.107E+05	4.217E+00	1.709E-02	8.344E-02	-0.025	-0.000
62	2.071E+02	6262.7	2.071E+04	1.784E+13	4.963E-08	2.768E-01	2.121E+05	5.623E+00	9.177E-02	1.468E-01	0.023	0.000
63	2.122E+02	6602.3	2.121E+04	3.454E+13	4.818E-08	4.927E-01	2.131E+05	7.499E+00	2.693E-01	2.105E-01	-0.017	-0.013
64	2.161E+02	6893.1	2.161E+04	5.797E+13	4.695E-08	7.898E-01	2.140E+05	1.000E+01	4.954E-01	2.330E-01	-0.039	0.001
65	2.196E+02	7116.2	2.196E+04	8.404E+13	4.616E-08	1.123E+00	2.147E+05	1.334E+01	6.685E-01	2.184E-01	0.064	0.000
66	2.230E+02	7306.3	2.229E+04	1.135E+14	4.559E-08	1.508E+00	2.155E+05	1.778E+01	7.584E-01	2.136E-01	-0.067	-0.001
67	2.264E+02	7485.1	2.263E+04	1.485E+14	4.511E-08	1.980E+00	2.162E+05	2.371E+01	8.200E-01	2.088E-01	-0.030	0.007
68	2.299E+02	7646.5	2.299E+04	1.875E+14	4.476E-08	2.525E+00	2.170E+05	3.162E+01	8.687E-01	1.945E-01	0.031	-0.001
69	2.337E+02	7796.3	2.336E+04	2.310E+14	4.452E-08	3.156E+00	2.178E+05	4.217E+01	9.013E-01	1.823E-01	-0.189	0.001
70	2.377E+02	7940.3	2.376E+04	2.804E+14	4.435E-08	3.902E+00	2.187E+05	5.623E+01	9.235E-01	1.754E-01	0.315	-0.000
71	2.420E+02	8085.4	2.419E+04	3.387E+14	4.422E-08	4.821E+00	2.197E+05	7.499E+01	9.412E-01	1.653E-01	-0.391	0.000
72	2.466E+02	8218.4	2.465E+04	4.009E+14	4.422E-08	5.842E+00	2.208E+05	1.000E+02	9.509E-01	1.688E-01	0.621	0.041

4.3 Calculating a synthetic spectrum with SYNTHE

Once an ATLAS9 or ATLAS12 model has been computed, a synthetic spectrum can be produced with SYNTHE for any specified set of abundances. In the example below we compute a model spectrum that includes the subordinate Na I lines near 5680 Å, with Na enhanced by +0.4 dex relative to the Solar-scaled abundance patterns and a microturbulent velocity of 2 km/s:

```
> from ispy3 import syntherk
>
> atoms, abun = ['Na'], [+0.4]
> vturb = 2.0
>
> syntherk.synbeg(5675., 5690., 0.01, initdir='/Volumes/ramdisk')
> syntherk.synthespec(-0.5, 't4200g200m050.A9', 't4200g200m050.asc',
>       atoms=atoms, abun=abun, vturb=vturb, initdir='/Volumes/ramdisk')
```

The procedure has two steps: first, synbeg() must be called to define the wavelength range and spectral resolution. This will also copy the relevant data from the line lists (atomic and molecular) and stage the data in initdir. The second step is the actual call to synthespec(), in which the logarithmic baseline scaling of the abundances is the first argument (−0.5) and any offsets in the abundances of specific elements are given in the arrays atoms and abun. Hence, in this example most elements will have [X/H]= −0.5, except sodium with [Na/H]= −0.5 + 0.4 = −0.1, or [Na/Fe]=+0.4. The model atmosphere is read from t4200g200m050.A9 and the synthetic spectrum will be stored in t4200g200m050.asc. The first few lines of the output file are:

```
# ISPy3 0.90.0
# SYNTHE
# atomdir = /Users/soeren/cats/linelists/atoms/ssl/SYNTHE/
# moldir  = /Users/soeren/cats/Kurucz/molecules/20apr2016/
# atmname = t4200g200m050.A9
# TEFF   4200.GRAVITY  2.000
# TITLE   [-0.5] VTURB=2.0 L/H=1.25 NOVER NEW ODF                               A
5675.0073  0.9792  4.9447e+05
5675.0171  0.9752  4.9247e+05
5675.0269  0.9692  4.8941e+05
5675.0371  0.9612  4.8537e+05
5675.0469  0.9528  4.8115e+05
...
```

After the ISPy3 version number, the second line indicates the code used to calculate the spectrum (here SYNTHE). The atomdir and moldir lines give the directories from which the atomic and molecular line data were read, as specified by the variables with the same names in syntherk.py. The actual model spectrum has three columns: The air wavelength (in Å), followed by the normalised spectrum, and finally the flux in physical units.

4.4 Interpolating in the MARCS grid

From the user's perspective, the procedure for obtaining a model by interpolation in the MARCS grid is (by design) closely analogous to that for computing an ATLAS model:

```
> from ispy3 import marcs
>
> m, teff, logg = -0.5, 4200., 2.00
> mass = 1.0
> marcs.intpatm(teff, logg, m, mass, 't4200g200m050')
```

The intpatm() function requires the mass as a fourth argument. However, this is currently ignored and a mass of 1 $M_{\odot}$ will be assumed for the spherical models no matter what is specified. The function also accepts the vturb argument, but this is also currently ignored and a microturbulent velocity of 2 km/s is assumed for spherical models while 1 km/s is assumed for plane-parallel models. Since no new model is computed, the call to intpatm() is very fast. The interpolated model is stored as t4200g200m050.marcs. The header indicates that the model was interpolated from the grid of spherical models, followed by the 56 layers of the atmosphere with a depth scale defined at 5000 Å. The last few lines list the models that were selected from the grid for the interpolation. Since there are three parameters, eight models are needed.

Interpolated `MARCS` model:

```
'sphINTERPOL'  56   5000.  2.00 0 0.00
 -3.9868  2800.20  -3.9141   1.5463   2.0000    0.116831E+13  -5.0000
 -3.8625  2825.70  -3.8243   1.6180   2.0000    0.116800E+13  -4.8727
 -3.7439  2853.77  -3.7290   1.6921   2.0000    0.116766E+13  -4.7455
 -3.6302  2883.44  -3.6302   1.7679   2.0000    0.116732E+13  -4.6182
 -3.5204  2913.77  -3.5300   1.8447   2.0000    0.116696E+13  -4.4909
 -3.4142  2944.50  -3.4295   1.9216   2.0000    0.116660E+13  -4.3636
 -3.3112  2975.55  -3.3295   1.9979   2.0000    0.116624E+13  -4.2364
 -3.2107  3006.66  -3.2306   2.0732   2.0000    0.116588E+13  -4.1091
 -3.1123  3037.69  -3.1333   2.1472   2.0000    0.116553E+13  -3.9818
 -3.0156  3068.53  -3.0379   2.2198   2.0000    0.116517E+13  -3.8545
 -2.9201  3099.15  -2.9444   2.2910   2.0000    0.116482E+13  -3.7273
 -2.8253  3129.45  -2.8531   2.3608   2.0000    0.116447E+13  -3.6000
 -2.7309  3159.28  -2.7640   2.4294   2.0000    0.116413E+13  -3.4727
 -2.6365  3188.70  -2.6769   2.4969   2.0000    0.116379E+13  -3.3455
 -2.5419  3217.41  -2.5922   2.5635   2.0000    0.116345E+13  -3.2182
 -2.4465  3244.94  -2.5103   2.6293   2.0000    0.116311E+13  -3.0909
 -2.3505  3272.35  -2.4294   2.6945   2.0000    0.116278E+13  -2.9636
 -2.2537  3299.59  -2.3498   2.7593   2.0000    0.116245E+13  -2.8364
 -2.1556  3325.73  -2.2725   2.8236   2.0000    0.116212E+13  -2.7091
 -2.0562  3351.27  -2.1966   2.8878   2.0000    0.116179E+13  -2.5818
 -1.9555  3376.32  -2.1219   2.9519   2.0000    0.116146E+13  -2.4545
 -1.8534  3401.07  -2.0482   3.0159   2.0000    0.116112E+13  -2.3273
 -1.7499  3425.70  -1.9750   3.0799   2.0000    0.116079E+13  -2.2000
 -1.6451  3450.68  -1.9019   3.1438   2.0000    0.116045E+13  -2.0727
 -1.5391  3476.47  -1.8282   3.2077   2.0000    0.116011E+13  -1.9455
 -1.4320  3503.59  -1.7534   3.2715   2.0000    0.115977E+13  -1.8182
 -1.3241  3532.95  -1.6768   3.3348   2.0000    0.115942E+13  -1.6909
 -1.2154  3565.25  -1.5976   3.3977   2.0000    0.115908E+13  -1.5636
 -1.1064  3601.11  -1.5155   3.4597   2.0000    0.115874E+13  -1.4364
 -0.9974  3642.45  -1.4290   3.5207   2.0000    0.115840E+13  -1.3091
 -0.8886  3689.59  -1.3386   3.5803   2.0000    0.115807E+13  -1.1818
 -0.7804  3744.03  -1.2435   3.6383   2.0000    0.115773E+13  -1.0545
 -0.6729  3806.63  -1.1443   3.6946   2.0000    0.115741E+13  -0.9273
 -0.5666  3879.03  -1.0411   3.7490   2.0000    0.115709E+13  -0.8000
 -0.4615  3962.30  -0.9348   3.8017   2.0000    0.115677E+13  -0.6727
 -0.3575  4058.02  -0.8266   3.8530   2.0000    0.115645E+13  -0.5455
 -0.2545  4167.54  -0.7179   3.9034   2.0000    0.115613E+13  -0.4182
 -0.1521  4292.54  -0.6097   3.9537   2.0000    0.115581E+13  -0.2909
 -0.0498  4435.26  -0.5022   4.0045   2.0000    0.115547E+13  -0.1636
  0.0527  4598.03  -0.3930   4.0565   2.0000    0.115511E+13  -0.0364
  0.1557  4784.16  -0.2704   4.1096   2.0000    0.115473E+13   0.0909
  0.2593  4996.95  -0.1083   4.1614   2.0000    0.115434E+13   0.2182
  0.3645  5240.27   0.1182   4.2074   2.0000    0.115398E+13   0.3455
  0.4717  5518.41   0.4101   4.2439   2.0000    0.115368E+13   0.4727
  0.5812  5836.58   0.7493   4.2704   2.0000    0.115344E+13   0.6000
  0.6920  6202.24   1.1171   4.2885   2.0000    0.115328E+13   0.7273
  0.8028  6586.93   1.4700   4.3005   2.0000    0.115316E+13   0.8545
  0.9128  6908.37   1.7391   4.3091   2.0000    0.115307E+13   0.9818
  1.0228  7150.16   1.9280   4.3164   2.0000    0.115298E+13   1.1091
  1.1333  7345.00   2.0716   4.3231   2.0000    0.115291E+13   1.2364
  1.2443  7521.30   2.1963   4.3298   2.0000    0.115283E+13   1.3636
  1.3561  7681.17   2.3052   4.3368   2.0000    0.115274E+13   1.4909
  1.4689  7828.77   2.4022   4.3440   2.0000    0.115266E+13   1.6182
  1.5826  7969.60   2.4920   4.3517   2.0000    0.115256E+13   1.7455
  1.6973  8105.58   2.5762   4.3600   2.0000    0.115246E+13   1.8727
  1.8128  8238.64   2.6565   4.3689   2.0000    0.115234E+13   2.0000
 s4000_g+2.0_m1.0_t02_st_z-0.50_a+0.20_c+0.00_n+0.00_o+0.20_r+0.00_s+0.00.mod


 s4000_g+2.0_m1.0_t02_st_z-0.50_a+0.20_c+0.00_n+0.00_o+0.20_r+0.00_s+0.00.mod


 s4000_g+2.5_m1.0_t02_st_z-0.50_a+0.20_c+0.00_n+0.00_o+0.20_r+0.00_s+0.00.mod


 s4000_g+2.5_m1.0_t02_st_z-0.50_a+0.20_c+0.00_n+0.00_o+0.20_r+0.00_s+0.00.mod


 s4250_g+2.0_m1.0_t02_st_z-0.50_a+0.20_c+0.00_n+0.00_o+0.20_r+0.00_s+0.00.mod


 s4250_g+2.0_m1.0_t02_st_z-0.50_a+0.20_c+0.00_n+0.00_o+0.20_r+0.00_s+0.00.mod


 s4250_g+2.5_m1.0_t02_st_z-0.50_a+0.20_c+0.00_n+0.00_o+0.20_r+0.00_s+0.00.mod
```

```
 s4250_g+2.5_m1.0_t02_st_z-0.50_a+0.20_c+0.00_n+0.00_o+0.20_r+0.00_s+0.00.mod


Interpolation point : Teff=   4200.  logg= 2.00  z= -0.50
 Optimized interpolation applied for standard composition models
```

4.5 Calculating a synthetic spectrum with TurboSpectrum

To compute a model spectrum with `Turbospectrum` for the `MARCS` model obtained above, the procedure is again largely equivalent to that for `SYNTHE`:

```
> from ispy3 import turbospec
>
> atoms, abun = ['Na'], [+0.4]
> vturb = 2.0
>
> turbospec.turboinit(5675., 5690., 0.01, initdir='/Volumes/ramdisk')
> turbospec.turbospec(-0.5, 't4200g200m050.marcs', 't4200g200m050marcs.asc',
>       atoms=atoms, abun=abun, vturb=vturb, initdir='/Volumes/ramdisk')
```

Recall, however, that `Turbospectrum` has its own set of line lists and other input data. One limitation is that `turboinit()` does not support the `elements` and `molecules` arguments that can be specified when calling `synbeg()` to include specific elements and molecules in the spectral synthesis.

4.6 Calculating equivalent widths with WIDTH9

The Kurucz `WIDTH9` code is quite quirky and can be used in many different ways. In `ISPy3`, the function `kurucz.calcew()` uses a modified version of `WIDTH9` to compute the equivalent width of a specified spectral line for a range of input abundances, which are given relative to the abundance of the corresponding element as specified in the header of the `ATLAS` model.

Naturally, an input line list is required. The format is the same as for the Kurucz atomic line list, but it is not necessary to include the full Kurucz list. A point worth noting is that while the wavelengths are specified in nm in the line list, the wavelength of the line must be given in Å when calling `calcew()` (as for other `ISPy3` functions). If we are interested in the Na I lines near 5680 Å, we might select the following entries:

```
568.2633 -0.700 11.00   16956.172  0.5 3p  2P      34548.766  1.5 4d  2D      0.00  0.00 -6.96KP   2 7  0 0.000  0 0.000    0    0              0    0    0
568.8193 -1.406 11.00   16973.368  1.5 3p  2P      34548.766  1.5 4d  2D      0.00  0.00 -6.96NIS3 2 7  0 0.000  0 0.000    0    0              0    0    0
568.8205 -0.452 11.00   16973.368  1.5 3p  2P      34548.731  2.5 4d  2D      0.00 -5.68 -6.96NIS3 2 7  0 0.000  0 0.000    0    0              0    0    0
```

When calling `calcew()`, the element is specified according to the same notation used in the Kurucz list, i.e. `ZZ.ii` where `ZZ` is the atomic number and `ii` is the degree of ionisation. Hence, for neutral sodium this would be `11.00`. To calculate the equivalent width of the Na 5682.633 Å line for five abundances for the `ATLAS9` model computed above, `calcew()` would then be called as follows:

```
> from ispy3 import syntherk
>
> elem = 11.00
> lambda = 5682.633
> result = syntherk.calcew('t4200g200m050.A9',elem, lambda,
      5, -1.0, 0.5, lines='na.lst')
> print(result['abund'])
[-7.21, -6.71, -6.21, -5.71, -5.21]
> print(result['ew'])
[68.73, 108.80000000000001, 146.78, 189.14999999999998, 249.88]
```

The results are returned in a `dictionary` structure with the keys `result['abund']` (the abundances for which equivalent widths are computed), `result['ew']` (the equivalent widths), `result['logew']` (the log of the equivalent widths), `result['depth']` (the `log(RHOX)` corresponding to an optical depth of unity for the line), `result['resid']` (the relative residual flux at the line centre) and `result['contin']` (the continuum flux).

In the example, the equivalent widths are computed for five abundances, starting from -1.0 dex below the Na abundance in the `ATLAS9` model in steps of 0.5 dex. Since the model was computed for [m/H]=-0.5 and the reference Solar abundance of Na (defined in `absetup.py`) is $\log n(\mathrm{Na}) = -5.71$, the equivalent widths are then computed for $\log n(\mathrm{Na}) = -5.71 - 0.5 - 1.0, -5.71-0.5-0.5, \ldots, -5.71-0.5+1.0 = -7.21, -6.71, \ldots, -5.21$. These values are confirmed by printing out `result['abund']`, and the corresponding equivalent widths are 68.73 mÅ, 108.80 mÅ, ..., 249.8 mÅ. Kurucz's version of `WIDTH9` supports a maximum of 9 individual abundances, but this has been increased to 99 in the modified version used by `ISPy3`.

4.7 Assigning a τ_{500} depth scale to ATLAS models

In the `ATLAS` models, the depth is parameterised as a column density `RHOX`, while the `MARCS` models use the continuum optical depth at 500 nm, τ_{500}. The function `kurucz.tau500()` can be used to generate a table with τ_{500} vs. RHOX for an `ATLAS` model. This is done by running the `ATLAS9` code with the line opacity disabled, stopping after one iteration so that the continuum opacities are recomputed but no modifications made to the model. The procedure amounts to a single call to `tau500()`.

```
> from ispy3 import kurucz
> kurucz.tau500('t4200g200m050.A9','t4200g200m050.tau500')
```

This is very fast, since no further iterations are needed and no line opacities are included in the calculation. A few lines of the output file are listed below. The `TAUNU` column contains the optical depth at 500 nm (as indicated in the header).

```
TEFF   4200.  GRAVITY 2.00000 LTE
      WAVELENGTH  500.000   FREQUENCY 5.995850E+14

        RHOX        TAUNU      ABTOT      ALPHA       BNU        SNU        JNU       JMINS       HNU
   1  1.037E-02  4.839E-06  4.668E-04  9.935E-01  1.937E-08  1.570E-06  1.581E-06  1.092E-08  9.971E-07  6.116E-01 -3.387E+12
   2  1.369E-02  6.391E-06  4.677E-04  9.919E-01  2.196E-08  1.569E-06  1.581E-06  1.252E-08  9.971E-07  7.024E-01 -4.202E+12
   3  1.782E-02  8.324E-06  4.687E-04  9.900E-01  2.481E-08  1.566E-06  1.581E-06  1.547E-08  9.971E-07  8.696E-01 -5.161E+12
   4  2.302E-02  1.077E-05  4.699E-04  9.877E-01  2.810E-08  1.562E-06  1.581E-06  1.912E-08  9.971E-07  1.077E+00 -6.301E+12
```

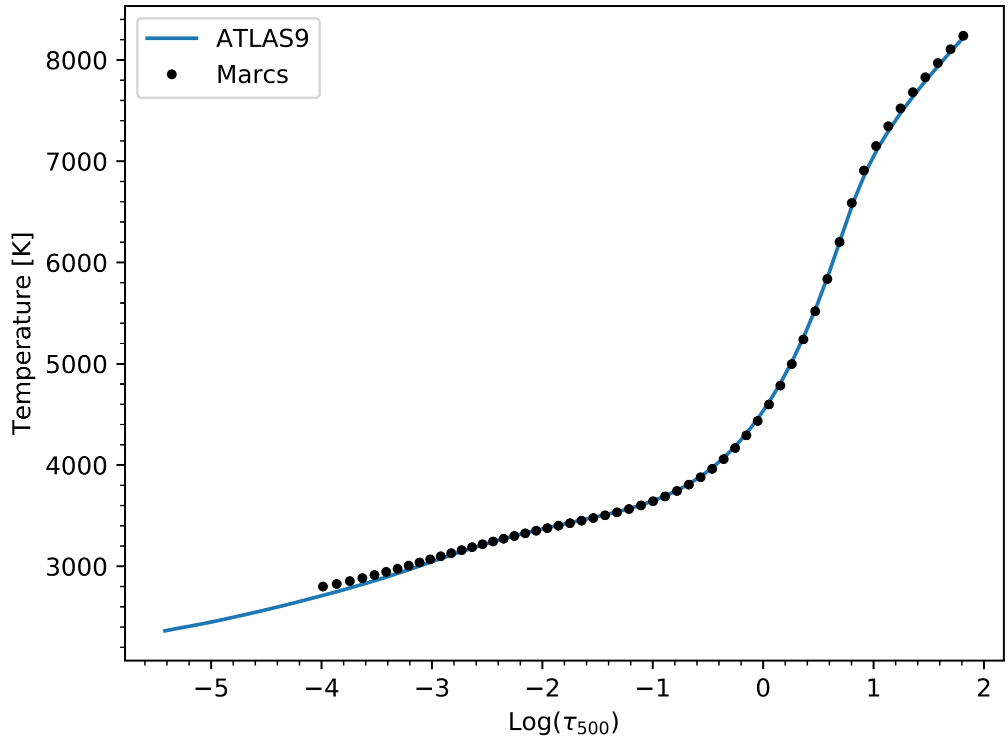

Figure 2: Comparison of `ATLAS9` and `Marcs` models for identical physical parameters ($T_{\rm eff}$ = 4200 K, $\log g$ = 2.0, [m/H]= −0.5.

```
 5  2.970E-02  1.391E-05  4.712E-04  9.849E-01  3.197E-08  1.558E-06  1.581E-06  2.346E-08  9.971E-07  1.326E+00 -7.687E+12
 ...
71  2.460E+02  5.170E+01  2.955E+00  1.857E-03  9.303E-05  9.303E-05  9.302E-05 -2.457E-09  1.455E-07 -8.708E+02 -3.688E+16
72  2.508E+02  6.709E+01  3.509E+00  1.825E-03  9.866E-05  9.866E-05  9.866E-05 -1.520E-09  1.221E-07 -6.394E+02 -3.737E+16
```

The `ATLAS` and `MARCS` models can now be compared directly, as shown in Fig. 2 for the same physical parameters as in Fig. 1. As can be seen, the temperature structures of the models are quite similar, although the `ATLAS9` model extends to lower optical depths.

4.8 Writing output to a log file

Many of the functions in `ISPy3` can use the `Python logging` module to write diagnostic output to a log file. To make use of this, the log needs to be configured with a call to `basicConfig()`:

```
> import logging
> logging.basicConfig(filename='ispy3.log', format='%(message)s',
      level=logging.INFO)
```

This enables logging, and output will automatically be written to (in this case) `ispy3.log`. `ISPy3` makes use of two levels of logging messages, `INFO` and `DEBUG`, where the latter produces more detailed information.

5 Determining abundances from spectral fitting

While the functions discussed above can be called separately by the user, they are also used by higher-level functions in `ISPy3` to derive abundances from fits to observed spectra of simple stellar populations (SSPs). As for single stars, the procedure involves two steps: First, the model atmospheres must be defined, and then the spectral fitting can be done.

5.1 Setting up the model atmospheres

To model the integrated-light spectrum of a simple stellar population, `ISPy3` needs a list of stellar parameters for the individual spectra that will be co-added. Usually this is stored in a regular text file, which can be read into a data structure in the format used by other tasks with the `abutils.rdphys()` utility function:

```
> from ispy3 import abutils
> stelpar = abutils.rdphys('hrd.txt')
```

The first few lines of the file `hrd.txt` might look as follows:

```
 MASS   TEFF    LOGG     RSTAR   WEIGHT   LOGVT SYNT ID
 0.513  3950.0  4.7355    0.510  1.625e+04 -0.301  MT  ISO
 0.547  4091.7  4.7010    0.548  1.396e+04 -0.301 A9S  ISO
 0.585  4288.4  4.6705    0.586  1.324e+04 -0.301 A9S  ISO
 0.622  4536.3  4.6504    0.619  1.151e+04 -0.301 A9S  ISO
```

The first line is a header that allows `rdphys()` to parse the remaining lines in the file. Only the columns `TEFF` (effective temperature), `LOGG` (surface gravity), `RSTAR` (radius of the star in Solar radii), and `WEIGHT` (the weight of the bin) are required. The logarithm of the microturbulent velocity (in km/s) can be specified in the `LOGVT` column; if this column is not included then `rdphys()` will assign the value `None` and the microturbulence must then be specified in some other way. The `SYNT` column specifies the combination of model atmospheres and spectral synthesis codes to be used for the modelling, with the following allowed values: `A9S` = `ATLAS9+SYNTHE` (default), `A12S` = `ATLAS12+SYNTHE`, `MT` = `MARCS+Turbospectrum`, and `A9T` = `ATLAS9+Turbospectrum`. The `A9T` combination requires the 2019 version of Turbospectrum, with the binaries `babsma_lu.19` and `bsyn_lu.19`. Note that `SYNT`, like the other parameters, is specified individually for each bin, so that it is straight forward to model different bins with different model atmospheres and spectral synthesis codes. The `MASS` column contains the mass, which is not used by `rdphys()`. Any other column headings are ignored. In the example above, the `ID` column is used to indicate that the bins come from an isochrone.

The `rdphys()` utility has an option `teffsort` whose default value is `True`. This means that the entries read from the input file will be sorted according to their temperature, and the atmospheres and spectra computed in that order. Since the cooler models and synthetic spectra usually take longer to compute, sorting the input list in this way makes the computation more efficient. The user should be aware, however, that this means that the ordering of the models stored on disc will usually be different from that in which they appear in the input list.

The model atmospheres can now be computed with a call to `abutils.hrd2atm()`:

```
> atoms = ['O','Ne','Mg','Si','S','Ar','Ca','Ti', 'Na']
> abun = [0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3]
> abutils.hrd2atm(stelpar, -0.8, atoms=atoms, abun=abun,
   tmproot='/scratch/soeren', odfs='A')
```

In this example, the model atmospheres will be computed for a baseline scaling of the abundances of [m/H]=−0.8 relative to the Solar composition. Abundances of individual elements are here specified with the `atoms` and `abun` variables, although this will not have much of an effect on the atmospheres unless models are computed with `ATLAS12`. The `odfs='A'` option, however, is passed on to `mkodf()` and specifies that α-enhanced ODFs will be used for the `ATLAS9` models. The `hrd2atm()` routine will run the atmosphere calculations for the different HRD bins in parallel and at the end store the models in files named `star0000`, `star0001`, etc, with a filename extension according to the type of atmosphere (e.g. `star*.marcs` or `*.A9`, etc.). The various temporary files that are generated while computing each model are stored in subdirectories under `tmproot` (here `/scratch/soeren`), which must exist before starting the procedure; for each model a separate temporary subdirectory with a unique name will be created and removed again once the calculation is complete, and only the final models are kept.

5.2 Carrying out the fitting

Chemical abundances can now be determined from an integrated-light spectrum by letting `ISPy3` vary the input abundances used in the spectral modelling until the best match to the observations is obtained. The `fitabun.py` module contains two functions for this purpose: `fitabun.fit1par()` is used to fit for the abundance of one element, while `fitabun.fitnpar()` can fit for the abundances of multiple elements simultaneously. The `abfit.py` module contains a utility method `fit1p()`, which uses `fitabun.fit1par()` to fit for the abundance of a single element but in addition allows more efficient managing of the input/output data, especially when multiple fits are to be carried out.

5.2.1 General aspects of the fitting procedure

To determine how well the SSP model spectrum matches the data, the continuum levels of the two spectra must be matched. If the data were properly flux calibrated, the matching would amount to a simple scaling of either the data or the model. In practice, achieving a sufficiently accurate flux calibration is often difficult, particularly for echelle spectra. Instead of a single value, `ISPy3` therefore applies a wavelength-dependent scaling $S(\lambda)$, which is defined by fitting a polynomial or a spline function to the ratio of the observed spectrum $F_{\rm obs}(\lambda)$ and the SSP model $F_{\rm syn}(\lambda)$.

$$S(\lambda) = \mathrm{fit}(F_{\rm obs}/F_{\rm syn}) \tag{3}$$

The observed spectrum and the error spectrum $E_{\rm obs}$ are then scaled by the fit,

$$F_{\rm obs,scl} = F_{\rm obs}/S \tag{4}$$

$$E_{\rm obs,scl} = E_{\rm obs}/S \tag{5}$$

and the χ^2 of the fit is evaluated as

$$\chi^2 = \sum_{i=1}^{n} w_i \left(\frac{F_{\rm obs,scl}(i) - F_{\rm syn}(\lambda_i)}{E_{\rm obs,scl}(i)} \right)^2 \tag{6}$$

where the sum is over n pixels in the observed spectrum with wavelengths λ_i and the w_i are user-specified weights.

The procedure can be customised via several variables defined in `fitabun.py`. The choice of continuum scaling function is controlled by `fitabun.CFITFNC`, which can be either `'poly'` or `'spline'`. The order of the scaling function is defined by `CFITORD` in both cases, with a value of 3 corresponding to a cubic spline. The choice `fitabun.CFITFNC='poly'` and `fitabun.CFITORD=0` would correspond to a scaling by a single numerical value, `CFITORD=1` would allow a linear dependence on wavelength, etc. For `CFITFNC='spline'`, the variable `NKNOTS` further defines the number of interior knots for the spline.

The pixels to include in the continuum scaling can be specified in various ways:

Leave out pixels that lie below some fraction of the continuum level: The variable `CFRAC` specifies that only pixels with values above a certain fraction of the "true" continuum level should be included when fitting the scaling function. How to define the "true" continuum is, however, a non trivial matter. In `ISPy3` this is done by finding the maximum pixel value within a wavelength range of `CDLAM` Å, centred on each pixel. This can be defined based on either the model spectrum (`CSEL='model'`) or the data (`CSEL='data'`). If the 'data' option is selected, then there is a risk that outlying pixel values will artificially boost the continuum level, but if one has a very good quality, high S/N spectrum this option may still be useful. Using the 'model' option avoids problems associated with noise, but is susceptible to imperfections in the synthetic spectra (such as missing lines). The default, `CFRAC=0`, implies that this scheme is disabled altogether.

Rejecting pixels where the fit is poor: A second scheme relies on rejection of pixel values that have a poor fit. After scaling the data, the standard deviation of the data-model difference is computed, and pixels that deviate by more than `CSIGP` standard deviations in the positive direction and more than `CSIGM` standard deviations in the negative direction are rejected. The continuum scaling is then recomputed, taking into account only the remaining pixels, and the procedure is repeated `CITER` times. The default, `CITER=1`, means no iterations are performed and this scheme is disabled. The asymmetric rejection limits are allowed on the grounds that outliers may be more likely in one direction than the other (for example, a significant number of lines may be missing from the line list). On the other hand, one has to pay careful attention to systematic effects if the rejection limits are very asymmetric.

Specifying the continuum regions explicitly: The third option is to provide a list of continuum flags explicitly when calling the `fit1par()` or `fitnpar()` functions, described below. The `cont` option is given in the form `[ [lam1, lam2, ...], [c1, c2, ...]  ]` where a value < 0.5 for each flag `c1, c2, ..` specifies that the corresponding wavelength is excluded, while a value of > 0.5 specifies that the corresponding wavelength is included when fitting the continuum. `ISPy3` will interpolate in these arrays to determine the flags at the wavelengths of the input data, hence the wavelengths of the `cont` array do not need to match those of the data.

5.2.2 Fitting a single element with `fit1par()`

The `fit1par()` task uses a simple Golden Section search to search for the best-fitting abundance of a single element, within some specified range. The uncertainty range is found by varying the abundance until the χ^2 of the fit has increased by unity relative to the χ^2 of the best fit.

The input to `fit1par()` is an observed spectrum which must be shifted to the rest frame, as well as the same set of stellar parameters used to set up the model atmospheres. The function `abutils.hrd2spec()` is used to compute model spectra for each HRD bin and co-adding them to produce the SSP models.

To determine the abundance of sodium from a fit to a spectrum stored in the text file `spectrum.txt`, relative to a baseline scaling of -0.8 dex relative to the Solar abundances, we can call `fit1par()` as follows:

```
> from ispy3 import fitabun
> import numpy as np
> fitabun.SPECRNG = [5677., 5695., 0.01]
> pfix = ['LogVT','LogZ','O','Ne','Mg','Si','S','Ar','Ca','Ti']
> vfix = ['USER', -0.8,  0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3]
> data = np.loadtxt('spectrum.txt',usecols=(0,1,2,3))
> stelpar = abutils.rdphys('hrd.txt')
> vfit,sigbst,vp,vm, chsq = fitabun.fit1par(data, stelpar, ['Na'], [-1.0, 1.0],
             pfix=pfix, vfix=vfix, sigsm=0.23, calcerr=True, cont=None)
```

The variable `fitabun.SPECRNG` is one of several global variables in `fitabun.py` that control the way in which the fitting is done (see Sec. 6). By setting it to `[5677., 5695., 0.01]`, we specify that the fit should be carried out for wavelengths between 5677 Å and 5695 Å and that the synthetic spectra should initially be computed at a resolution of 0.01 Å. They will then be smoothed as specified by the `sigsm` argument and rebinned to the resolution of the observations.

The first argument to `fit1par()` is a list of `[[lambda1, flux1, err1, weight1], [lambda2, flux2, err2, weight2], ..]` values for each pixel, which are typically read in from a text file. The wavelengths should be specified in Å measured in air. In the example we use `np.loadtxt()` to read the data from the file `spectrum.txt`, assuming that the radial velocity correction has already been applied. A typical use of the `weight` column is to exclude certain pixels from the fit by setting `weight=0`. The same could, of course, be achieved by setting the `err` to some very large value.

Next follow the stellar parameters. It is critically important that these are *exactly* the same, and appear in the same order, as those used to compute the model atmospheres, as `fit1par()` will use the corresponding already existing model atmospheres for the spectral synthesis.

The third argument specifies the element to be fitted (here 'Na'), followed by the range of abundances to search, relative to the baseline scaling (`LogZ`). Here, we specify a range of $[-1, +1]$ dex with respect to the scaled baseline abundances, [m/H]= -0.8. The `pfix` argument specifies which parameters are to be kept fixed in the fit and `vfix` their corresponding values. In addition to the abundances of specific elements, these parameters also include `LogVT` (the microturbulent velocity) and `LogZ` (the baseline scaling of the abundances relative to Solar). In the example, `LogVT` is set to `USER`, meaning that the microturbulent velocities specified in the `stelpar` array will be used. A single numerical value can also be given, which will then be used for the modelling of all the spectra. `LogZ` is set to -0.8, so the baseline scaling of the abundances is -0.8, while the alpha-elements are enhanced by 0.3 dex. The `sigsm` parameter specifies the dispersion of the Gaussian kernel used to smooth the model spectrum (in Å) and `calcerr` indicates whether the one-sigma errors should be calculated. If `sigsm` is set to a range, `fit1par()` will search for the best-fitting smoothing.

The `cont` argument can be used to specify the sampling points for the continuum scaling. These are defined separately from the weights and errors defined in the input spectrum, the idea here being that one might want to exclude certain regions of the spectrum when matching the continuum scaling. This could be the case, for example, if the line list is known to be inadequate in some parts of the spectrum.

The `fit1par()` routine returns the best-fit value of the fitted parameter (`vfit`), the best smoothing (`sigbst`), the positive and negative errors (`vp`, `vm`), and the reduced χ^2 of the fit (`chsq`). At the end, we can inspect the results:

```
> print(vfit)
> 0.38699100866857605
> print(sigbst)
> 0.23
> print(vp, vm)
> 0.0107421875 -0.015625
> print(chsq)
> 3.013505275006322
```

where we see that the best-fitting sodium abundance is increased by 0.39 dex with respect to the baseline (LogZ= -0.8), i.e. [Na/m]= $+0.387^{+0.011}_{-0.016}$. Since `sigsm` was specified as a single value, that same value is returned.

In addition to the elements, `fit1par()` can also fit for the baseline `LogZ`, which is treated as a special "element". To fit for the baseline and the smoothing, one would specify:

```
> pfix = ['LogVT','O','Ne','Mg','Si','S','Ar','Ca','Ti']
> vfix = ['USER', 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3]
> vfit,sigbst,vp,vm, chsq = fitabun.fit1par(data, stelpar, ['LogZ'],
      [-3.0, 1.0], pfix=pfix, vfix=vfix, sigsm=[0.01, 1.0], calcerr=True)
```

This will then solve for a scaling of the reference abundances in the range $[-3.0, 1.0]$ with respect to Solar composition, with the alpha-elements enhanced by 0.3 dex, and for the best-fitting Gaussian smoothing with a dispersion between 0.01 and 1.0 Å. Clearly, it would be non-sensical to specify LogZ (or any other element) both in the `pfix` array and as a parameter to be fitted, but `fit1par()` has sufficient faith in the user that it does not explicitly check this.

In addition to the return values, `fit1par()` produces a file with the spectrum and the best fit, called `fitabun.txt`. This can then be inspected to check the quality of the fit. This file will be overwritten every time `fit1par()` is called, so remember to make a copy with a different name if you want to keep it.

Similarly to the calculation of model atmospheres, the intermediary files for the spectral synthesis for each HRD bin are stored in temporary directories. These will be created as sub-directories relative to `absetup.TMPROOT`, which must exist before the calculation is started.

5.2.3 Fitting multiple elements with `fitnpar()`

The `fitnpar()` function can fit for abundances of multiple elements. Starting from an initial estimate of the abundances, the downhill simplex method of Nelder & Mead will be employed to search for the best fit. An example call to fit for the abundances of Fe and Na might look as follows:

```
> from ispy3 import abutils
> from ispy3 import fitabun
> import numpy as np
> import logging
>
> logging.basicConfig(filename='fitn.log',format='%(message)s', level=logging.INFO)
>
> stelpar = abutils.rdphys('hrd_t10m070p04_hb0104.txt')
>
> fitabun.SPECRNG = [5677., 5695., 0.01]
> pfix = ['LogVT','LogZ','O','Ne','Mg','Si','S','Ar','Ca','Ti']
> vfix = ['USER', -0.8, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3]
> data = np.loadtxt('spectrum.txt',usecols=(0,1,2,3))
>
> vfit = fitabun.fitnpar(data, stelpar,
>                ['Fe','Na'], [0., 0],
>                pfix=pfix, vfix=vfix, sigsm=0.23)
>
> print(vfit)
```

Here, the elements to be fitted are specified in the third argument (`['Fe','Na']`) and the initial guesses are specified in the fourth argument. The other variables are the same as in the `fit1par()` example. Note that in this example we have also enabled logging to the file `fitn.log`, so that progress of the fitting can be followed by inspecting the log file. At the end, the last few lines in the log file will contain information about the best fit:

```
FITNPAR (5677.000 - 5695.000):
    Final Fit:
     Fe : +0.008
     Na : +0.386
    Broadening = 0.230
    Reduced chi-square = 2.987
    Number of iterations = 21
```

This shows the wavelength range, the best-fit abundances and smoothing, as well as the reduced χ^2 of the best fit and the number of iterations. The best-fit values are also returned by `fitnpar()`:

```
> print(vfit)
[0.0078125, 0.38593749999999997]
```

where we see that [Fe/m]= 0.008 and [Na/m]= +0.386. Reassuringly, the best-fit [Na/m] is very similar to the value obtained from the `fit1par()` fit above, and only a small adjustment of the Fe abundance relative to the baseline is required, so that we find [Na/Fe]=+0.378. While `fitnpar()` can fit for the broadening by giving a range (as for `fit1par()`), the best-fit value is not returned and must therefore be looked up in the log file.

`fitnpar()` allows several elements to be fit together as a group. To this end, a list of elements can be specified instead of a single element. To fit for Fe and the alpha-elements, one could specify

```
> palpha = ['O','Ne','Mg','Si','S','Ar','Ca','Ti']
> pfix = ['LogVT','LogZ']
> vfix = ['USER', -0.8]
> vfit = fitabun.fitnpar(data, stelpar, ['Fe',palpha], [0., 0.3],
            pfix=pfix, vfix=vfix, sigsm=0.23)
```

Again, one has to be careful to avoid elements being specified both as parameters to be fit and to be kept fixed.

A limitation of `fitnpar()` is that it does not provide a way to estimate the errors on the fitted parameters, so it is up to the user to do this in some other way.

5.2.4 Finding and applying the radial velocity shift

There are various ways in which the radial velocity of a spectrum may be measured and applied. `ISPy3` includes a function called `deltarv()` which will search for the radial velocity offset in a given interval that gives the best match between a model and observed spectrum. This usually works best if a rough correction (to within a few km/s) has already been applied, for example based on identification of a few prominent lines.

Suppose we have estimated the radial velocity shift of a spectrum to be about −10 km/s. We can then use `abutils.rvcorrect()` to apply this correction to the wavelength scale of the spectrum, stored in the same format used by `fit1par()` and `fitnpar()`:

```
> import numpy as np
> from ispy3 import abutils
> rv = -10
> data = np.loadtxt('n0104_1d_redu.txt',usecols=(0,1,2,3))
> data2 = abutils.rvcorrect(data, rv)
```

To refine the radial velocity estimate, we can then use `deltarv()` to search for the best match in a range of ±10 km/s:

```
> from ispy3 import fitabun
> lam1, lam2 = 5000., 5100.
> fitabun.SPECRNG = [lam1, lam2, 0.01]
> drv = fitabun.deltarv(data2, stelpar, -0.8, pfix, vfix,
     sigsm=0.2, drvmax=10.)
```

As with `fit1par()` and `fitnpar()`, the model atmospheres must already be available and must match the stellar parameters specified in `stelpar`. In general, it is wise to repeat the procedure for several wavelength windows and use an average as the final estimate of the radial velocity shift. The `deltarv()` function is a fairly "quick-and-dirty" hack and you might well be able to code something much better yourself!

5.2.5 Integrated-light equivalent widths

The function `abutils.hrd2ew()` can be used to compute equivalent widths for lines in an integrated-light spectrum. Similarly to `abutils.hrd2spec()`, the model atmospheres must be computed before the call to `abutils.hrd2ew()`. To calculate the equivalent width for the Na I 5682 Å line, the function would then be called as follows:

```
> ew5682 = abutils.hrd2ew(stelpar, 11.00, 5682.633, lines='na.lst',
    minlog=-1.0, dablog=0.5, nablog=5, logvt='USER')
```

As in the `calcew()` example, the equivalent widths are computed for five abundances (`nablog=5`), starting from −1.0 dex with respect to the abundance specified in the model atmospheres (`minlog=-1.0`) and in steps of 0.5 dex (`dablog=0.5`). The output is an array with the equivalent widths for each abundance value.

5.3 The `abfit.py` module

The `abfit.py` module provides a higher-level interface to the `fit1par()` function, intended to simplify the managing of the input/output data and streamline the procedure for carrying out multiple fits to a given spectrum. The module contains a single class, `abfit.setup()`, which initialises the setup for a sequence of fits for a single element and opens an output file for the results. Upon initialisation, the class returns an object with the associated method `fit1p()`, which can then be used to carry out multiple fits for the abundance of the element in different wavelength regions and neatly write the results to the output file. At the end, the output file is closed with a call to the `close()` method. A slightly modified version of the `fit1p()` method, called `fitZ()`, is optimised for the global metallicity fits. Information that is not specific to a particular element, such as the name of the input files, spectral resolution, radial velocity, the continuum definition file, etc., is defined separately via functions that are passed on to `abfit.setup()` at the time of initialisation. The idea behind this approach is to avoid, as much as possible, that the same information has to be specified in multiple files.

Here's a basic example to fit for the abundance of iron in two spectral bins:

```
> from ispy3 import abfit
> import setup
> logz = -0.8
> pfix = ['LogVT','LogZ','O','Ne','Mg','Si','S','Ar','Ca','Ti']
> vfix = ['USER',logz,  0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3]
> stelpar = setup.getsp()
> S = abfit.setup('Fe', pfix, vfix, stelpar, 'fitFe.out',
            setup.setfname, setup.calcrv, setup.calcsigsm, setup.fncont)
> S.fit1p('u', [5400., 5420.], 'fitFe540.txt','poly',2)
> S.fit1p('u', [5420., 5460.], 'fitFe542.txt','spline',3,nknots=3)
> S.close()
```

The initialisation method, `abfit.setup()`, requires several arguments to be specified in order to define the details of the fit: First, the element to be fitted must be given (here `'Fe'`), followed by the arrays `pfix` and `vfix` that define the parameters to keep fixed and their corresponding values. These will be passed on to `fit1par()`. The stellar parameters are given in the usual format, and are then followed by the name of the output file for the results (`'fitFe.out'`). After this follow a number of functions: `setfname(key)` should return the name of the file containing the input spectrum, as a function of a key is passed as the first argument when calling the fitting method `fit1p()`. In the example here, the spectra are stored in two files, `n104l.txt` and `n104u.txt`, so the key can be 'l' or 'u' and `setfname()` will construct the full file name accordingly. This is just a way of keeping the calls to `fit1p()` as concise as possible. The next argument, `calcrv(lambda)` should be a function that returns the radial velocity correction to be applied to the spectrum, which can be a function of the wavelength. Similarly, `calcsigsm(lambda)` should return the Gaussian broadening to be applied to the spectrum, again as a function of wavelength. Finally, `fncont()` should return the name of the file containing the continuum definition data. These functions are generally common to all elements that one might want to fit, and can therefore conveniently be defined in a separate file, here called `setup.py`:

```
from ispy3 import abutils

def getsp():
    stelpar = abutils.rdphys('hrd_t10m070p04_hb0104.txt')
    return stelpar

def calcrv(lamc):
    return -19.300
```

```
def calcsigsm(lamc):
    return lamc * 4.0e-5

def setfname(ordername):
    return 'n104'+ordername+'.txt'

def fncont():
    return 'cont.dat'
```

Note that a function to read the HRD data has also been defined here.

After initialisation, `abfit.setup()` returns an object that includes the `fit1p()` method. The first argument to this method is the key that defines the input file, followed by the wavelength range, the name of an output file for the spectral fit (which will be renamed from the `fitabun.txt` file produced by `fit1par()`) and the parameters defining the continuum scaling function. This can be either `poly` or `spline`, followed by the order of the fitting function (3 for a cubic spline) and, when using a spline, the number of internal knots. In the example, the abundance of iron is fitted in two spectral windows. At the end, the output file is closed with the `close()` method. This produces the output file `fitFe.out` with the results of the fit:

```
# ISPy3 0.90.1
# soeren@soerenl2.astro.ru.nl /Users/soeren/BTSync2/projects/Abundance/doc/n0104
# Python 3.7.3 Darwin 17.7.0 2020-02-06 11:28:32.050801
# Fitting: Fe
#  BestFit       Lam1    Lam2   Result  err+   err-    Chisq   (sig)
fitFe540.txt    5400.00 5420.00 -0.079 +0.006 -0.008   3.859
fitFe542.txt    5420.00 5460.00 -0.122 +0.005 -0.005   3.434
# Done at 2020-02-06 12:15:34.133749
```

A second method, `fitZ()` is very similar to `fit1p()`, but optimised to fit parameters over a broader wavelength range. The default values for the arguments to the method differ somewhat: the default range of the `fitrng` argument is $[-4.0, 0.5]$ (appropriate for the LogZ fits), and the `calcerr` argument is set to `False` as default. As one typical use of `fitZ()` is to determine the best-fitting broadening (by setting the `sigsm` argument to a range), the broadening applied to the spectrum will be written to the output file as an extra column (whether it was fitted or not).

5.4 Setting up batch jobs on the coma cluster at Radboud University

6 User configurable variables

Some of the modules in ISPy3 contain a variety of variables that can be modified by the user. Some of these may be set automatically from higher level routines.

absetup.py

`absetup.binpath`: A string pointing to the directory in which the external binaries are located.

`absetup.catpref`: This variable should point to the root directory for the various data files needed by the model atmosphere and spectral synthesis codes. More details are given in the descriptions of `ATLAS/SYNTHE` (sec. 3.1) and `Marcs/TurboSpectrum` (sec. 3.2).

`absetup.TMPROOT` is the name of a directory for storage of temporary files. The `SYNTHE` spectral synthesis code in particular relies heavily on reading and writing large amounts of temporary data, so for optimum performance it is important that this directory resides on a local disc that allows fast access to read/write operations.

`absetup.MAXPROC` is the maximum number of processes to be run in parallel.

`absetup.stdabun` contains the reference abundance scale (Grevesse & Sauval 1998). Negative numbers are logarithms of fractional number densities relative to the *total*, as per the convention in the `ATLAS` models.

fitabun.py

`fitabun.SPECRNG` = [5000., 5050., 0.01] # Spectral range and wavelength step size of synthetic spectra. This will be set when calling `fit1p()`, but will need to be specified explicitly before calling the `fitabun.fit1par()` or `fitabun.fitnpar()` functions directly.

`fitabun.EXPAND` = 1.0 # When computing the synthetic spectra, add `EXPAND` extra Å at both ends of the fitted wavelength range. This ensures that wings of lines beyond the fitted range are included in the modelling. It should hardly ever be necessary to change this.

`fitabun.CFITORD` = 3 # Order of continuum fitting function. Usually this will be set when calling `fit1p()`.

`fitabun.CFITFNC` = 'spline' # Type of function used to fit the continuum shape. Can be 'spline' or 'poly'. Will usually be set when calling `fit1p()`.

`fitabun.NKNOTS` = 5 # Number of knots when `CFITFNC = 'spline'`.

`fitabun.CFRAC` = 0.0 # Only use pixels with $F(\lambda)/F_{\max} >$ `CFRAC` when matching the continuum scaling of the spectra. $F_{\max}$ is evaluated over a wavelength range from $\lambda -$ `CDLAM`/2 to $\lambda +$ `CDLAM`/2. If `CFRAC` = 0 (default) then all pixels are used.

`fitabun.CDLAM` = 5.0 # Width of wavelength range for computing $F_{\max}$.

`fitabun.CITER` = 1 # Number of iterations for continuum fitting

`fitabun.CSIGP` = 2.0 # Upper clipping threshold for continuum fitting fitabun.CSIGM = 1.5 # Lower cliplling threshold for continuum fitting

fitabun.CSEL = 'MODEL' # Select continuum points based on 'MODEL' or 'DATA'

fitabun.NPINT = 1 # Number of interpolation points per pixel in the observed spectrum. The default value of 1 means that the synthetic spectrum is simply interpolated at the central wavelength of each pixel *after* smoothing to account for velocity broadening and instrumental resolution. If the observed spectra are severely undersampled, NPINT can be increased in order to sample the synthetic spectrum at more points within each pixel before rebinning to the resolution of the data.

fitabun.SMOOTHFNC = 'gaussian' # Type of profile used to smoothen the model spectra. Current options are 'gaussian' or 'uniform'.

kurucz.py

kurucz.LOGTAU0 = -6.875 # Logarithm of Rosseland mean optical depth of outermost layer for ATLAS models (both ATLAS9 and ATLAS12)

kurucz.DLOGTAU = 0.125 # Logarithmic step in Rosseland depth per layer (both ATLAS9 and ATLAS12)

kurucz.NDEPTH = 72 # Number of layers in atmosphere model (both ATLAS9 and ATLAS12). The default values of these variables will thus produce a model with a maximum Rosseland mean depth of 100.

kurucz.AFE = 'A' # Use alpha-enhanced ('A') or solar-scaled ('S') initial models when starting from the pre-existing set. This will be set automatically when model atmospheres are computed via abutils.hrd2atm(). At any rate, the value is usually not critically important since this only concerns the starting model for the iterations.

kurucz.A9_NREP = 1 # Number of times to repeat 15 extra ATLAS9 iterations. A model calculation always starts with 15 iterations, hence the default value implies a total of 30 iterations.

kurucz.A9_INIT_EXT = None # Filename extension for ATLAS9 initial models

kurucz.A9_FROMSCRATCH = False # Calculate ATLAS9 models from scratch instead of starting from a pre-existing model

kurucz.A12_NREP = 2 # Number of times to repeat 15 extra ATLAS12 iterations. A model calculation always starts with 15 iterations, hence the default value implies a total of 45 iterations.

kurucz.A12_INIT_EXT = None # Filename extension for ATLAS12 initial models. A typical use of this option would be to first compute a set of ATLAS9 models via a call to abutils.hrd2atm()), and then use these as starting points for ATLAS12 models. In this case, one would set kurucz.A12_INIT_EXT = '.A9'. The default value, None, implies that the initial model is selected from the pre-existing set.

syntherk.py

`syntherk.atomdir` # Directory with atomic line lists for `SYNTHE`

`syntherk.moldir` # Directory with molecular line lists for `SYNTHE`

`syntherk.preddir` # Directory with predicted lines for `SYNTHE`

7 Functions

Module kurucz.py

`mkodf(m,atoms=[],abun=[], odfs='S'):` Interpolate in the library of pre-computed opacity distribution functions (ODFs) to define an ODF for a specified metallicity *m*.
Parameters:

> `m`: The metallicity of the desired ODF in logarithmic units, relative to the Sun (i.e. `m` = 0 is Solar metallicity, `m` = −2 means 1/100 Solar metallicity, etc)
>
> `odfs='S'`: Composition of the ODF. Can be 'S' (Solar-scaled) or 'A' (Alpha-enhanced)
>
> `atoms=[], abun=[]`: ignored.

`mkatm(teff, logg, m, atmname, workdir='.', wait=True, atoms=[], abun=[], nlte=False, vturb=0.0, firstguess=None)`: Calculate an `ATLAS9` model atmosphere for specified physical parameters. Before calling `mkatm()`, an interpolated ODF must be generated via a call to `mkodf()`.
Parameters:

> `teff`: The effective temperature of the model (in K)
>
> `logg`: The logarithm of the surface gravity of the model (in cgs units)
>
> `m`: The metallicity of the model in logarithmic units relative to the Sun. More precisely stated, `m` is the logarithm of the default scaling factor for the abundances in the reference (Solar) abundance scale specified in `absetup.stdabun`.
>
> `atmname`: A string specifying the filename for the computed model. In fact, two output files will be produced, namely the model atmosphere itself (with an extension `'.A9'` appended to `atmname`) and a diagnostic output file produced by the `atlas9mem.exe` code with more details about the calculation (with extension `'.out'` appended to `atmname`) that can be useful for checking convergence of the models.
>
> `workdir='.'`: The directory in which the actual calculation is carried out. At the end of the calculation, only the atmosphere and diagnostic output file are copied back to the current directory. When multiple atmospheres are being computed in parallel by `ISPy3`, temporary directories will automatically be set up for each model to keep the files from being overwritten.

`wait=True`: Wait for the calculation to be completed? The default, `True`, means that the function will wait until the atmosphere has been computed. For `wait=False` the model calculation will be launched as a background process and `mkatm` will return a process ID that allows `ISPy3` to keep track of calculations running in parallel.

`atoms=[]`: An array of elements for which the composition is specified

`abun=[]`: The abundances for the elements specified in `atoms`, relative to the scaling factor `m`. While the line opacity is computed via the ODFs by `ATLAS9`, modifying the abundances of individual elements can still have a significant effect on an `ATLAS9` model. This is the case, for example, for elements such as Mg and Si that are significant electron donors and therefore affect the H^- opacity, which is a dominant continuum opacity source in solar-type and cooler stars.

`nlte=False`: Include NLTE effect in the model atmosphere calculation? Default is False, since the NLTE treatment in ATLAS9 is anyway not very realistic.

`vturb=0.0`: The microturbulent velocity, in km/sec. This is used to select the appropriate ODF. ODFs are computed for 0, 1, 2, 4, and 8 km/sec, of which the closest match will be selected.

`firstguess=None`: Explicitly specify an existing model to use as an initial guess. If `None`, a model from the pre-existing set will be selected.

`mkatm_a12(teff, logg, m, atmname, workdir='.', wait=True, atoms=[], abun=[], nlte=False, vturb=2.0, firstguess=None)`: Calculate an `ATLAS12` model atmosphere for specified physical parameters. Since `ATLAS12` uses opacity sampling, it is not necessary to first call `mkodf()`. The parameters have the same meaning as for the `ATLAS9` models (`mkatm()`), with a few differences in the detailed behaviour:

- The output model has extension `'.A12'` and the diagnostic output file is named by adding `'b.out'` to `atmname`.
- Any changes in the abundance patterns specified with `atoms=[]` and `abun[]` are fully accounted for in the model.
- The specified microturbulent velocity is taken into account self-consistently by the `atlas12.exe` code in the opacity calculations.

`tau500(atmname, tauname)`: Use `atlas9mem.exe` to generate a table with the continuum optical depth at 500 nm (τ_{500}) for an existing `ATLAS9` or `ATLAS12` model.

`calcodf(m, odfname, atoms=[], abun=[])`:: Use the `dfsynthe.exe` code to calculate ODFs for a specified composition. The `m`, `atoms`, and `abun` variables have the same meaning as described above. The ODFs will be computed for microturbulent velocities of 0, 1, 2, 4, and 8 km/sec for temperatures between 1995 K and 199526 K.

Module fitabun.py

`fit1par(specobs, stelpar, pfit, prange, pfix=[], vfix=[], tol=0.001, calcerr=True, sigsm=[0,1.0], cont=None)`:

```
fitnpar(specobs, stelpar, pfit, pinit, pfix, vfix, tol=0.001, sigsm=[0,1.0],
cont=None):
```

References

[1] Alvaro, R. & Plez, B., 1998, A&A, 330, 1109

[2] Castelli, F., & Kurucz, R. L., 2003, in: Modelling of Stellar Atmospheres, IAU Symp. 210

[3] Hernandez, S., Larsen, S. S., Trager, S. C., et al. 2017, A&A, 603, A119

[4] Hernandez, S., Larsen, S. S., Trager, S. C., et al. 2018, MNRAS, 473, 826

[5] Larsen, S. S., Brodie, J. P., & Strader, J., 2012, A&A, 546, A53

[6] Larsen, S. S., Brodie, J. P., Forbes, D. A., & Strader, J., 2014, A&A, 565, A98

[7] Larsen, S. S., Brodie, J. P., & Strader, J., 2017, A&A, 601, A96

[8] Masseron, T. 2008, *Interpolation of MARCS stellar atmosphere models*, Manual

[9] Nelder & Mead reference

[10] Plez, B., 1998, A&A 337, 495

[11] Plez, B., 2012, Astrophysics Source Code Library, record ascl:1205.004

[12] Sbordone, L. 2005, MSAIS 8, 61, Proceed. of ”ATLAS12 and related codes” workshop, editors P. Bonifacio and S. J. Adelman