PFANT Manual

Release 17.12.21

Julio Trevisan

Dec 21, 2017

CONTENTS

Welcome!

PFANT is a stellar spectral synthesis software for use in Astronomy.

INTRODUCTION

PFANT is a stellar spectral synthesis software written in Fortran.

The development started with M Spite *et al.* in France in 1967 (Figure 1). The code formulation is described in *(Barbuy 1982)*, *(Cayrel et al. 1991)*, *(Barbuy et al. 2003)*, and *(Coelho et al. 2005)*.

```
 |
 | 1967 -- FANTÔME    -- by M Spite et al.
 | 1982 -- FANTOMOL   -- B Barbuy included the computation of molecular lines,
 |                       dissociatipon equilibrium
 | 2003 -- PFANT      -- M-N Perrin: large wavelength coverage,
 |                       inclusion or hydrogen lines
 | 2015 -- PFANT      -- J Trevisan: source code conversion to Fortran 2003 and
→optimization;
 |                       Python interface, tools and API
t|
 V
```

Figure 1 - Summarized PFANT timeline *(Coelho et al. 2005)*

1.1 References

(Barbuy 1982) Barbuy, B. 1982, PhD Thesis, Université de Paris VII

(Cayrel et al. 1991) Cayrel, R., Perrin, M. N., Barbuy, B., & Buser, R. (1991). A grid of synthetic spectra for the determination of effective temperature, gravity and metallicity of F, G, and K stars. I-Description of the method. II-Application to 41 stellar spectra taken in the Basel field of SA 141. Astronomy and Astrophysics, 247, 108-129.

(Barbuy et al. 2003) Barbuy, B., Perrin, M.-N., Katz, D. et al. 2003, A&A, 404, 661

(Coelho et al. 2005) Coelho, P., Barbuy, B., Meléndez, J., Schiavon, R. P., & Castilho, B. V. (2005). A library of high resolution synthetic stellar spectra from 300 nm to 1.8 μm with solar and α-enhanced composition. Astronomy & Astrophysics, 443(2), 735-746.

INSTALLATION

To use PFANT, you will need to:

1. Download files

2. Compile the Fortran source code

3. Add `PFANT/fortran/bin` to your PATH

4. Install the "f311" Python package (http://github.com/trevisanj/f311) (recommended)

This section will take you through these steps.

Note: PFANT is platform-independent (it should work no any system if you can install the GNU Fortran Compiler), however only Debian-based Linux system is "supported" in the following instructions. Windows users will find some tips in a specific section below.

2.1 Installing required software

2.1.1 Standalone applications

Please install the following standalone applications on your system (no pain except for gfortran and make on Windows (see below)):

- gfortran (version 4.8 recommended; does **not** compile with version 4.4)

- make

2.2 Download files

2.2.1 Clone the GitHub repository:

```
git clone https://github.com/trevisanj/PFANT
```

This will create a directory named PFANT on your disk.

2.3 Compiling the Fortran source code.

Enter the following on your console to compile the Fortran source code:

```
cd PFANT
cd fortran
./make-linux.sh
```

This should create four executable binaries inside the directory *PFANT/fortran/bin*: `innewmarcs`, `hydro2`, `pfant`, `nulbad`.

2.4 Setting the paths

Depending on which shell your system uses, try one of the following:

Bash shell:

```
./add-path.py --bash
```

Tcsh shell:

```
./add-path.py --tcsh
```

This will automatically apply the path settings to your *home/.bashrc* or *home/.cshrc*.

Note: If the above does not work for you, manually add *PFANT/fortran/bin* to your system path.

2.5 Install `pyfant` Python package

Although PFANT contains standalone a set of tools for spectral synthesis, it is recommended to install the `pyfant` Python package to add running, editing, visualization and conversion capabilities around the Fortran core. Installation instructions are available at http://trevisanj.github.io/pyfant

2.6 Tips for windows users

2.6.1 gfortran and make on Windows

MinGW (http://sourceforge.net/projects/mingw/files/) is a convenient way to install the GNU Fortran compiler on Windows.

After installed, MinGW has its own package manager, named "MinGW Installation Manager". There, you will need to install at least the following packages: `mingw-developer-toolkit`, `mingw32-base`, `mingw32-gcc-fortran`, `msys-base`.

2.6.2 Compiling the source code on Windows

The source can be compiled using the CodeBlock Fortran IDE. The *PFANT/fortran* forder contains a CodeBlocks project named *PFANT-windows.cbp*.

QUICK START

Aims for this tutorial:

- calculate a synthetic spectrum;

- convolve with Gaussian functions of varying full-width-at-half-maximum (FWHM);

- visualize results.

3.1 Spectral synthesis from the command line

3.1.1 Short story

Here is the full command sequence:

```
mkdir mystar
cd mystar
copy-star.py
link.py
run4.py
plot-spectra.py --ovl flux.norm flux.norm.nulbad.0.120
```

3.1.2 Long story

Create a new directory

```
mkdir mystar
cd mystar
```

Gather input data

Input data consists of:

1. stellar parameters (temperature, chemical abundances etc.) and running settings (*e.g.*, calculation wavelength interval);

2. star-independent physical data: line lists, atmospheric model grid, partition functions etc. that are less likely to be modified. We refer to these as "common" data.

Stellar data and running settings

The following displays a menu allowing you to choose among a few stars:

```
copy-star.py
```

After running this, the following files will be copied into the "mystar" directory:

- "main.dat": main configuration (editable with `mained.py, x.py`)

- "abonds.dat": chemical abundances (editable with `abed.py, x.py`)

Common data

For these data, we will create links instead of copying the files, as the files are big and/or unlikely to change:

```
link.py
```

The following links should appear in your directory now:

- absoru2.dat

- atoms.dat

- grid.mod

- grid.moo

- hmap.dat

- molecules.dat

- partit.dat

Spectral synthesis pipeline

Spectral synthesis involves a few steps, as shown in Figure 3.1, and described in the next subsections.

```
+-------------------+     +-----------------+     +---------------+     +-----------+
| innewmarcs        |     | hydro2          |     | pfant         |     | nulbad    |
| ==========        |     | ======          |     | =====         |     | ======    |
|    interpolate the|     |           create| --> | calculate     | --> | convolve  |
|        atmospheric| --> | hydrogen lines  |     | synthetic     |     |      with |
|              model|     |         profiles|     | spectrum      |     | Gaussian  |
+-------------------+     +-----------------+     +---------------+     +-----------+
```
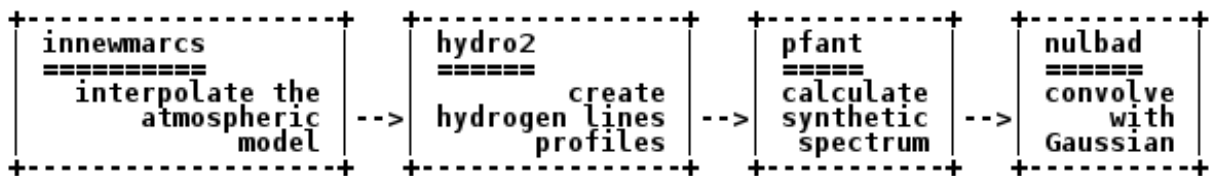
Figure 3.1: – PFANT spectral synthesis pipeline showing the Fortran program names and what they do.

Interpolate the stellar atmospheric model

This step takes a 3D grid of atmospheric models (usually a file named "grid.mod") and interpolates a new model given a certain point (temperature, gravity, metallicity) (specified in the main configuration file) contained within the limits of the grid.

```
innewmarcs
```

will create two files: "modeles.mod" and "modeles.opa".

Note: If the combination of (temperature, gravity, metallicity) is outside the limits of the grid (*e.g.*, if you choose star Mu-Leo), `innewmarcs` will refuse to interpolate. However, it can be forced to use the nearest points in the grid with:

```
innewmarcs --allow T
```

Create hydrogen lines profiles

```
hydro2
```

will create files such as: "thalpha" (Figure 5.4), "thbeta", "thgamma" etc.

Calculate synthetic spectrum

```
pfant
```

creates files:

1. "flux.spec": spectrum
2. "flux.cont": continuum
3. "flux.norm": normalized spectrum ((1) divided by (2))

To visualize these files:

```
plot-spectra.py flux.spec flux.cont flux.norm
```

will open a plot window (Figure 3.2).

Convolve synthetic spectrum with Gaussian function

The following will take the normalized spectrum from the previous step and convolve it with a Gaussian function of FWHM = 0.12, creating file "flux.norm.nulbad.0.120":

```
nulbad --fwhm 0.12
```

Plot spectra

```
plot-spectra.py --ovl flux.norm flux.norm.nulbad.0.120
```

opens a plot window where one can see how the spectrum looks before and after the convolution (Figure 3.3).

Running the four calculation steps at once

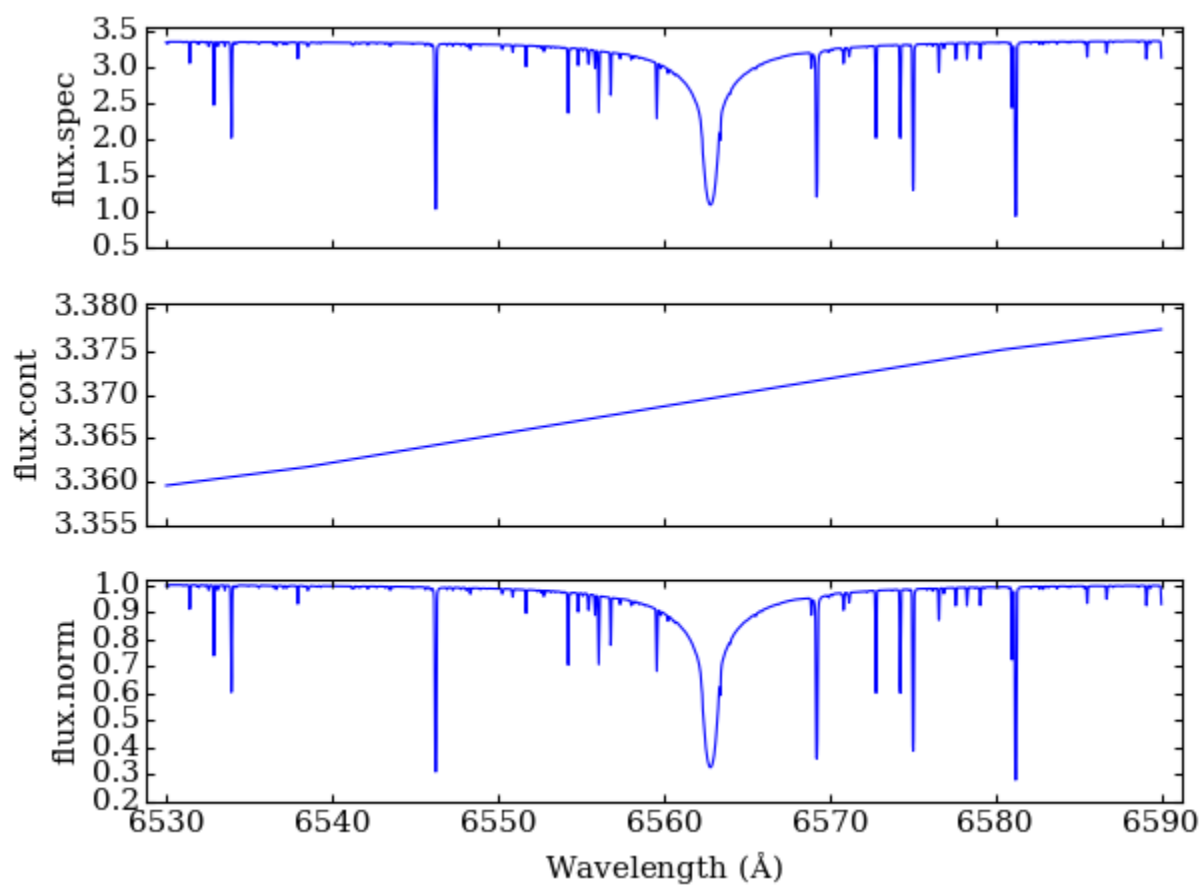The script `run4.py` is provided for convenience and will run all Fortran binaries in sequence.

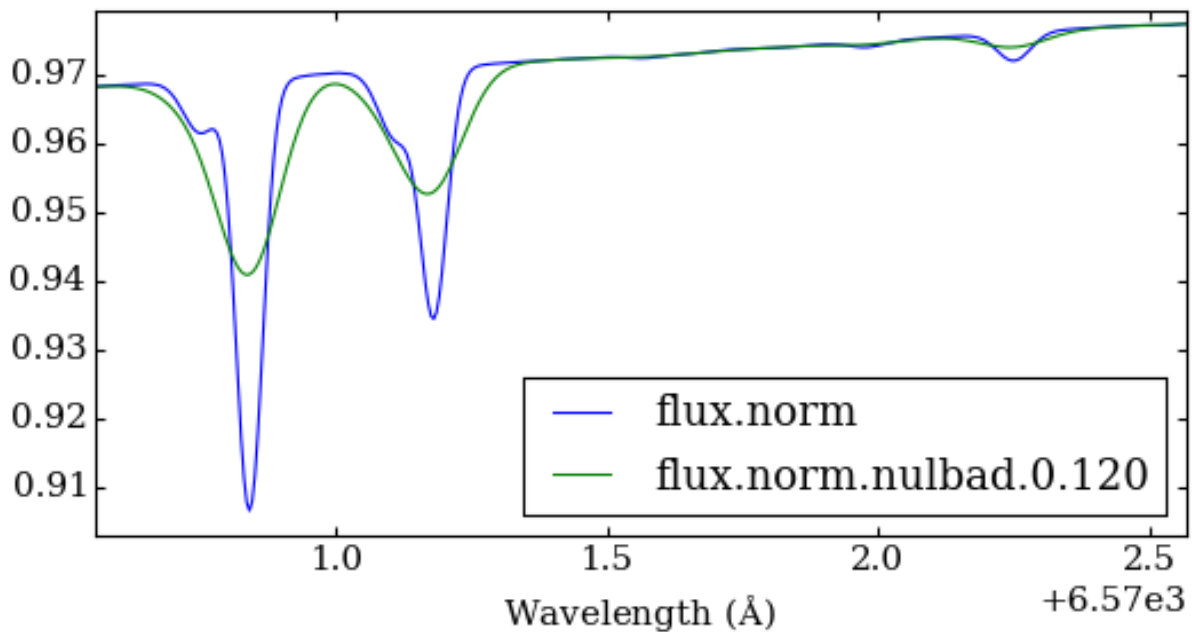Figure 3.2: – plots of three files generated by `pfant`.

Figure 3.3: – plot comparing spectra without and after convolution with Gaussian function (FWHM = 0.12)

```
run4.py --fwhm 0.12
```

Hint: The same command-line options available in the Fortran binaries are available in `run4.py`.

3.2 Spectral synthesis using the Graphical interface

Alternatively to the command line, you can use the "PFANT Launcher" (*x.py*) provided by the F311 project.

First it is necessary to create a new directory and gather the input data (as in the spectral synthesis from the command line above):

```
mkdir mystar
cd mystar
copy-star.py
link.py
```

Now you can invoke the "PFANT Launcher" application (Figure Figure 3.4):

```
x.py
```

Here is a suggested roadmap:

1. Change parameters in Tab 1/2/3 (Tab 4 is a different story)

2. Click on the "Submit single job" button: a new window named "Runnables Manager" opens
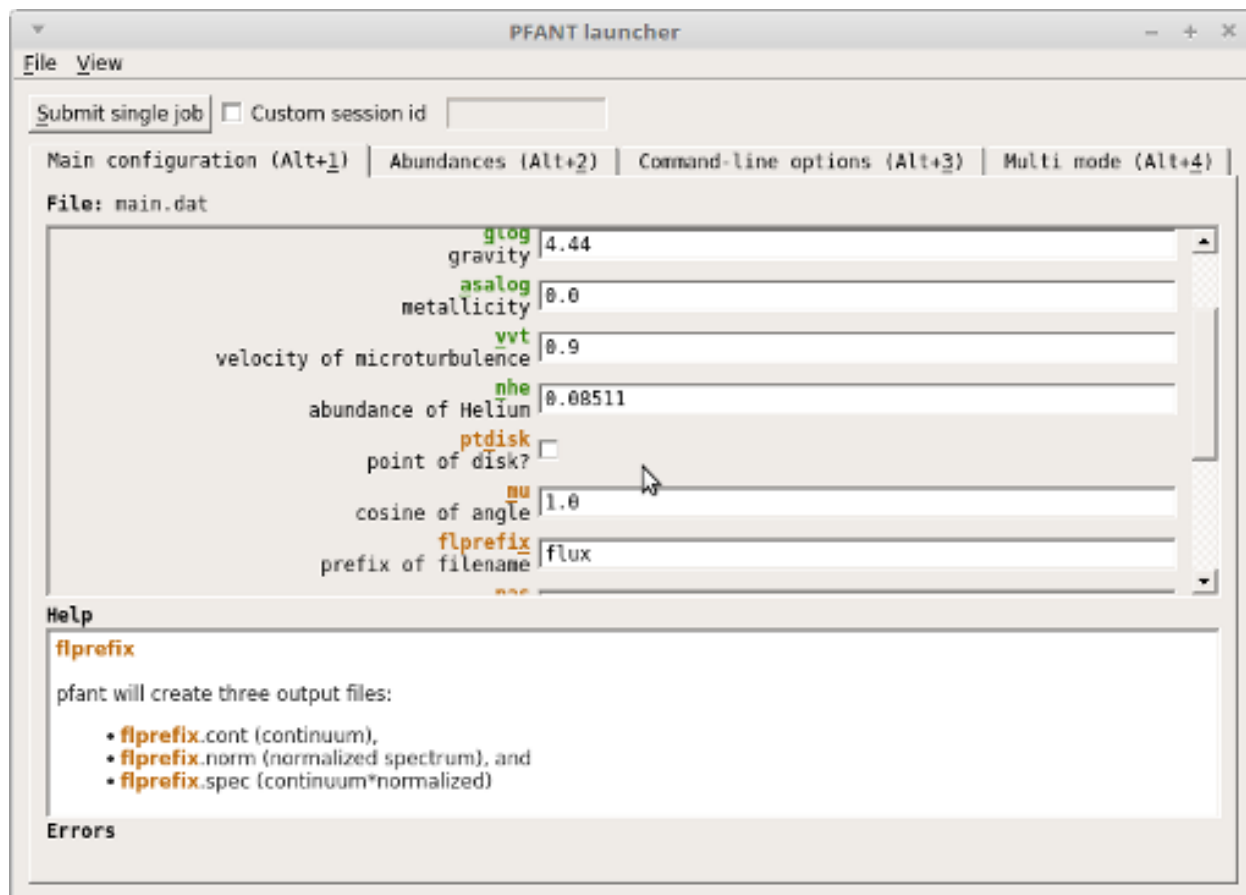
Figure 3.4: – Screenshot of the x.py application

3. When the "Status" column shows "nulbad finished", double-click on the table item: "PFANT Explorer" window opens

4. Double-click on "flux.norm": turns green (if wasn't so)

5. Double-click on "Plot spectrum": spectrum appears

3.3 Writing Python scripts with package `pyfant`

Python package `"pyfant` provides an API that allows one to perform spectral synthesis from Python code, manipulate PFANT-related data files, and more.

Here is a simple spectral synthesis example. The following code runs the Fortran binaries (`innewmarcs`, `hydro2`, `pfant`, `nulbad`) in a way that is transparent to the Python coder, and then plots resulting synthetic spectra (Figure 3.5):

```python
import pyfant
import f311
obj = pyfant.Combo()
obj.run()
obj.load_result()

# Plots continuum, spectrum, normalized in three sub-plots
f311.plot_spectra_stacked([obj.result["cont"],
                           obj.result["spec"],
                           obj.result["norm"]])

# Plots normalized unconvolved, normalized convolved spectra overlapped
f311.plot_spectra_overlapped([obj.result["norm"],
                              obj.result["convolved"]])
```
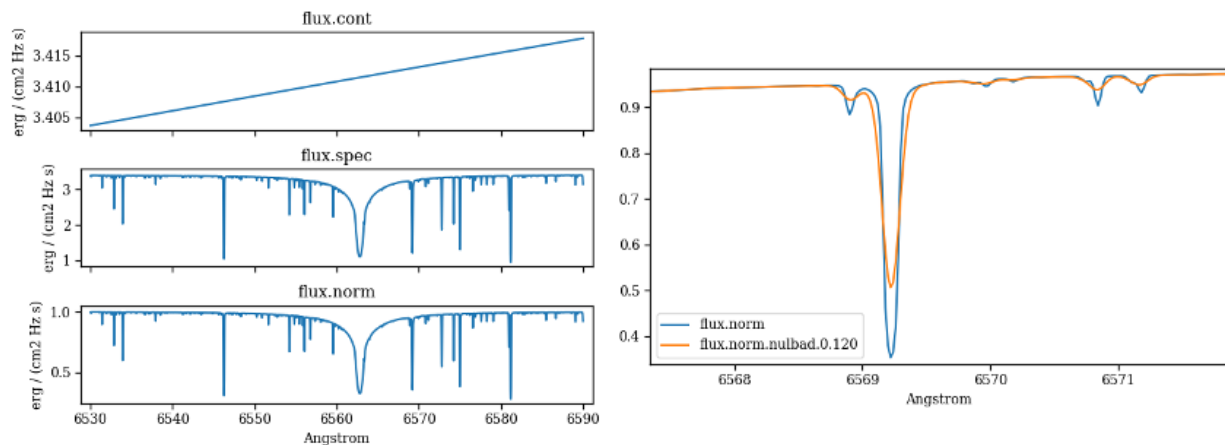

Figure 3.5: – Plots generated from code above.

More Python examples can be found at https://trevisanj.github.io/pyfant

MORE GUI

4.1 Editing input data files

4.1.1 Browse files with F311 Explorer

This application (Figure 4.1) allows you to navigate through your file system and visualize/edit files of various files, including spectra and most files used by PFANT. A list with all supported file types is available here
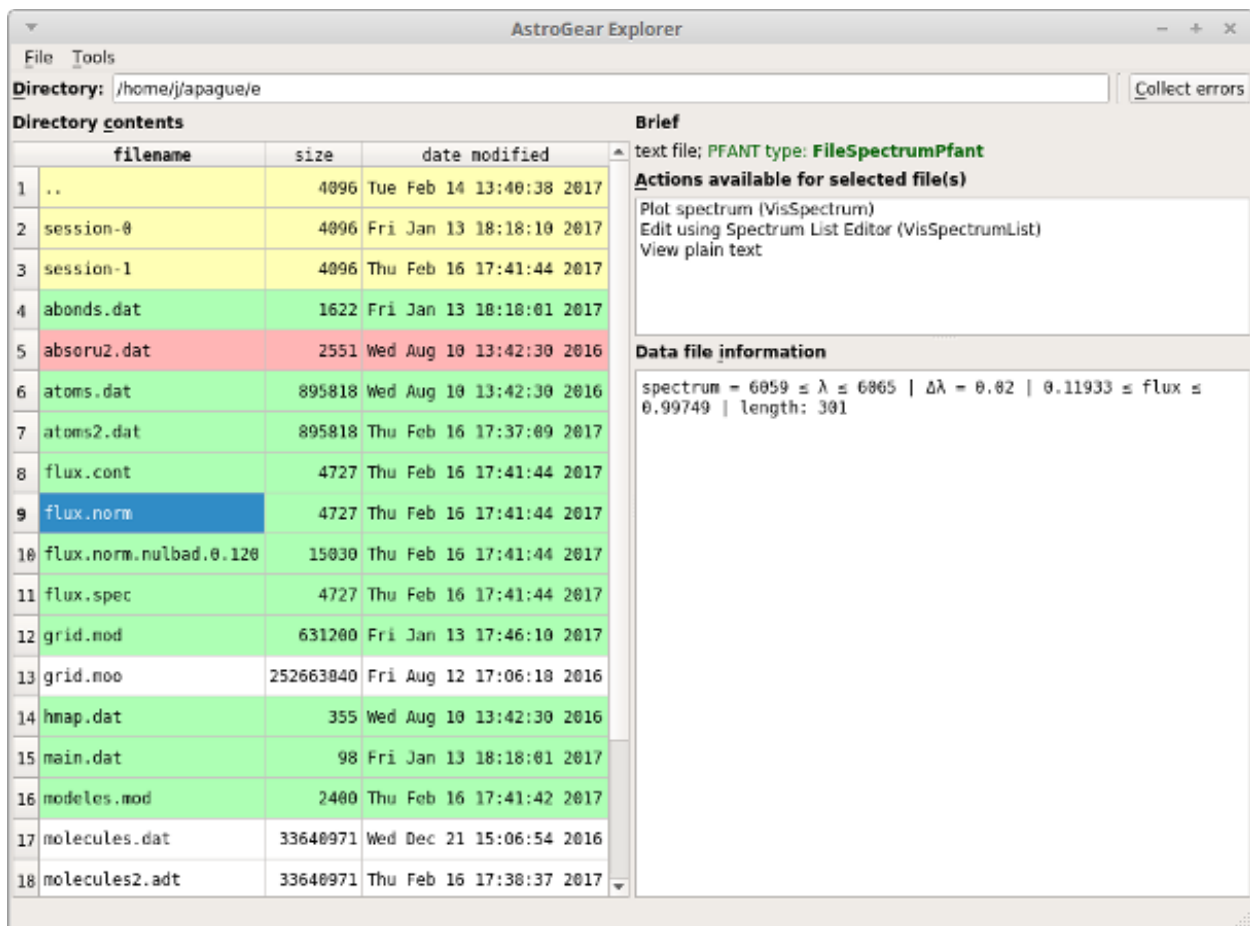
```
explorer.py
```


Figure 4.1: – `explorer.py` screenshot.

You can select several spectral files and plot them all at once (stacked in different sub-plots, or overlapped in a single plot).

4.1.2 Edit Atomic Lines file

First make a copy of file "atoms.dat" to leave the current one untouched.

```
copy atoms.dat atoms2.dat
```

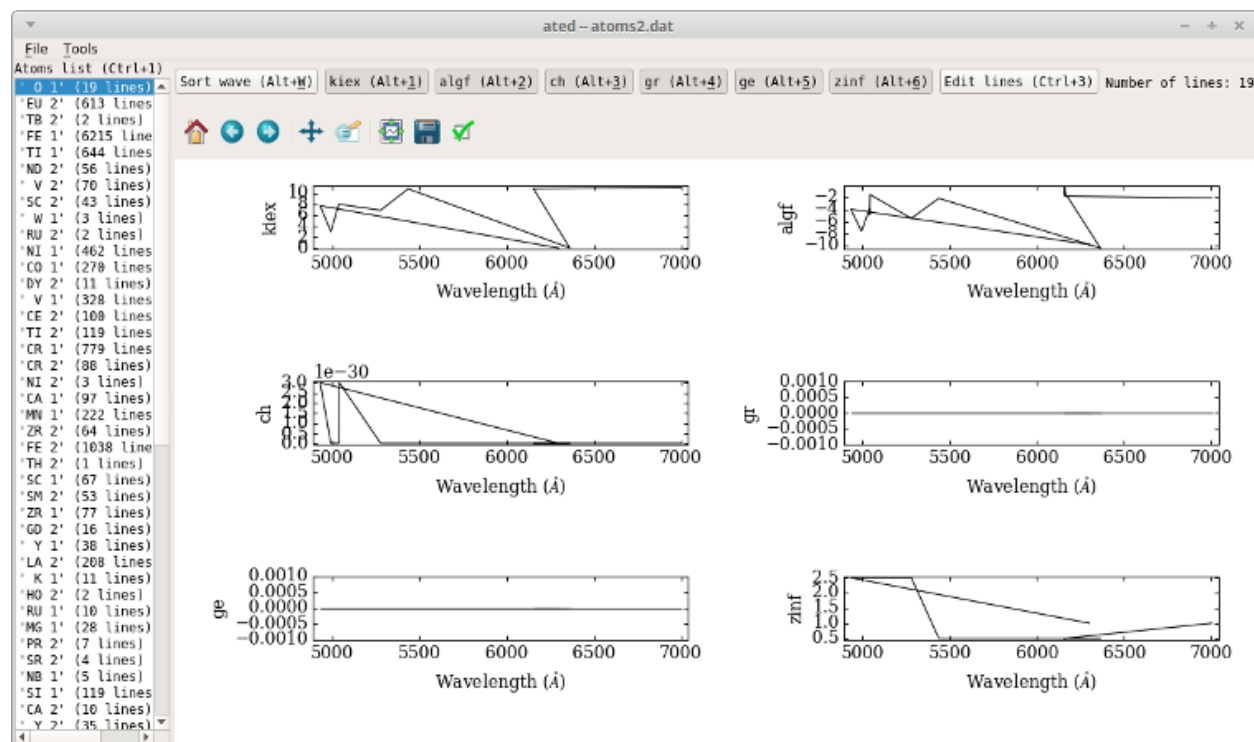Now open the Atomic Lines Editor (Figure 4.2):

```
ated.py atoms2.dat
```


Figure 4.2: – `ated.py` screenshot.

4.1.3 Edit Molecular Lines file

First make a copy of file "molecules.dat" to leave the current one untouched.

```
copy molecules.dat molecules2.dat
```

Now open the Molecular Lines Editor (Figure 4.3):

```
mled.py molecules2.dat
```

4.1.4 Other editors and tools

Check *Cheatsheet* for a complete list of applications related to PFANT spectral synthesis.
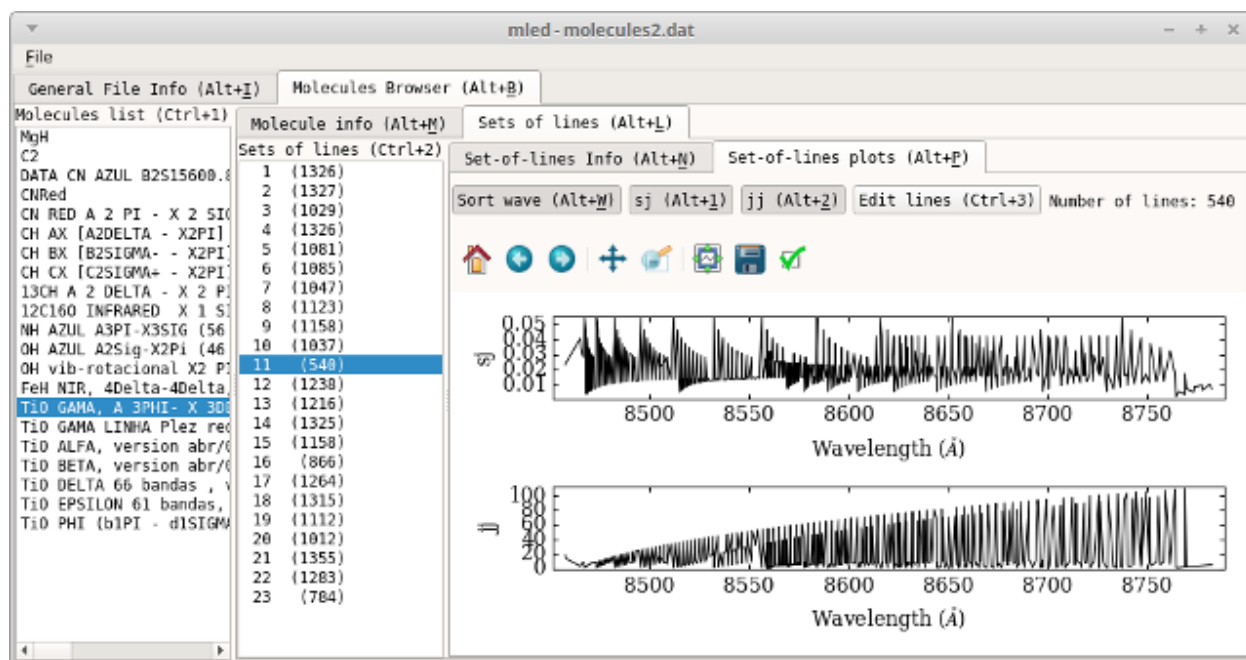
Figure 4.3: – `mled.png` screenshot.

REFERENCE

This section contains a more complete description of the PFANT pipeline and the files and file types involved.

5.1 Spectral synthesis pipeline

```
                              +---------------------------+---------------------main.dat
                              |                           |
                              v                           v  (H lines)
grid.mod        +-----------+                   +------+
    or  ---->|innewmarcs|-->modeles.mod--->|hydro2|<----------+
grid.moo        +-----------+                   +------+
             (interpolate                          |
              atm. model)                          |
                                                    v
                                                 thalpha
                                                 thbeta            absoru2.dat
                                                 thgamma           hmap.dat
                                                 thdelta
                                                 thepsilon
                         |                          |
            abonds.dat   |                          v
            dissoc.dat   +--------->+------+
              atoms.dat   (synthesis)|pfant|<----------+
          molecules.dat --------------->+------+<---------------------------+
             partit.dat                   |
                                          v
                                       flux.norm
                                       flux.spec
                                       flux.cont
                                          |
                                          v
                                       +------+
          (convolve spectrum)  |nulbad|<-----------------+
                                       +------+
                                          |
                                          v
                              flux.norm.nulbad.<fwhm>
```
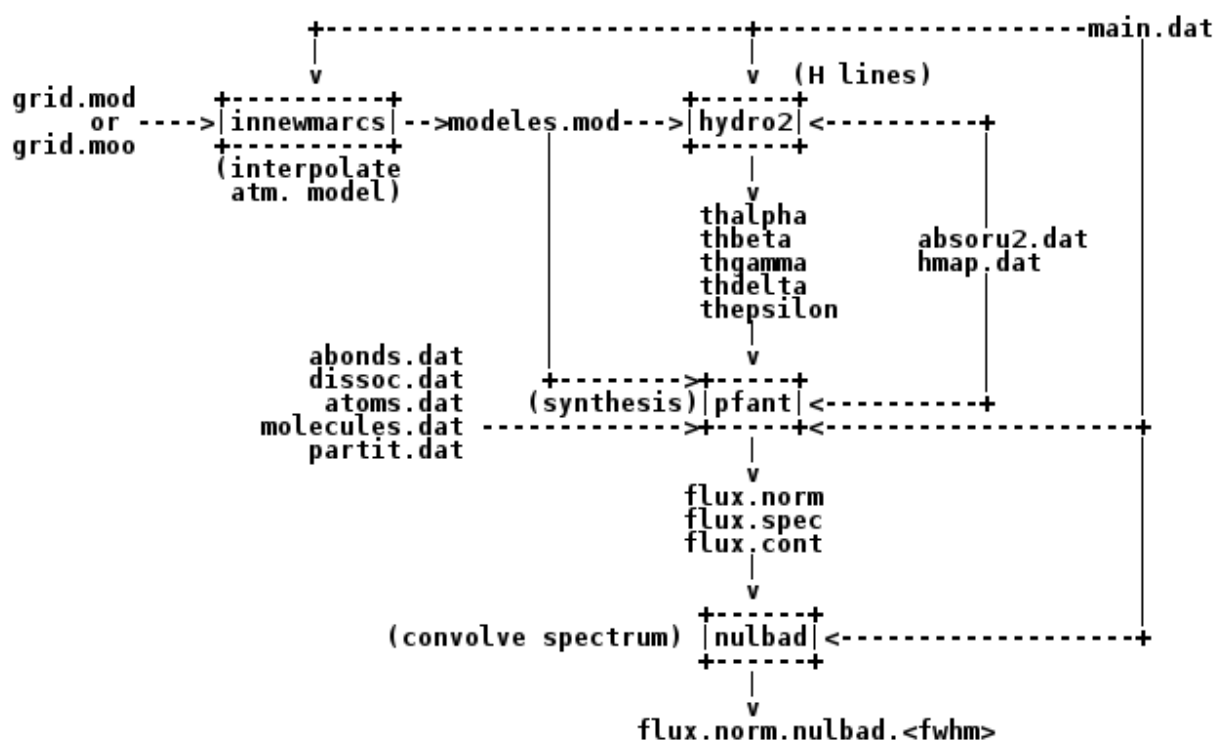
Figure 5.1: – Spectral synthesis pipeline - Fortran programs (boxes) and their input/output files.

5.2 Input/output data files

The different file types in the pipeline will be explained in the next subsections.

5.2.1 Stellar data and running settings

Table 2 – stellar and running settings data files. The "–options" column shows the command-line options that can be used to change the name for a particular file, *e.g.*, `run4.py --fn_main main-other.dat`.

Default name	–option	Description
main.dat	–fn_main	main configuration file containing all stellar parameters except abundances
abonds.dat	–fn_abonds	chemical abundances
dissoc.dat	–fn_dissoc	dissociation equilibrium data. This file is optional, and can be created using `abed.py` if needed

5.2.2 Common data files

Table 3 – Common data files.

Todo: fix table

Default name	–option	Description
absoru2.dat	–fn_absoru2	absorption info for continuum calculation.
atoms.dat	–fn_atoms	atomic line list
molecules.dat	–fn_molecules	molecular line list
hmap.dat	–fn_hmap	hydrogen line list
partit.dat	–fn_partit	partition functions
grid.mod or	–fn_modgrid	MARCS atmospheric model grid (models only)
grid.moo	–fn_moo	MARCS atmospheric model grid (models with opacities) (Figure 5.2)

5.2.3 Files created by the Fortran programs

Files created by `innewmarcs`

Table 4 – Files created by `innewmarcs`

Default name	command-line option	Description
modeles.mod	–fn_modeles	atmospheric model (binary file) (Figure 8A)
modeles.opa	–fn_opa	atmospheric model: opacities (MARCS ".opa" format) (Figure 8B,8C)

`innewmarcs` creates two separate files (Table 4). They are created separately for historical reasons. "modeles.opa" follows the same structure of ".opa" files downloaded from the MARCS website. "modeles.mod" does **not** follow the same structure of MARCS ".mod" files. Figure 8 illustrates the information contained in these files.

Files created by `hydro2`

`hydro2` creates a series of files named "thalha" (Figure 5.4), "thbeta", "thgamma", "thdelta", "thepsilon" etc. (the series of hydrogen lines is given in file "hmap.dat").
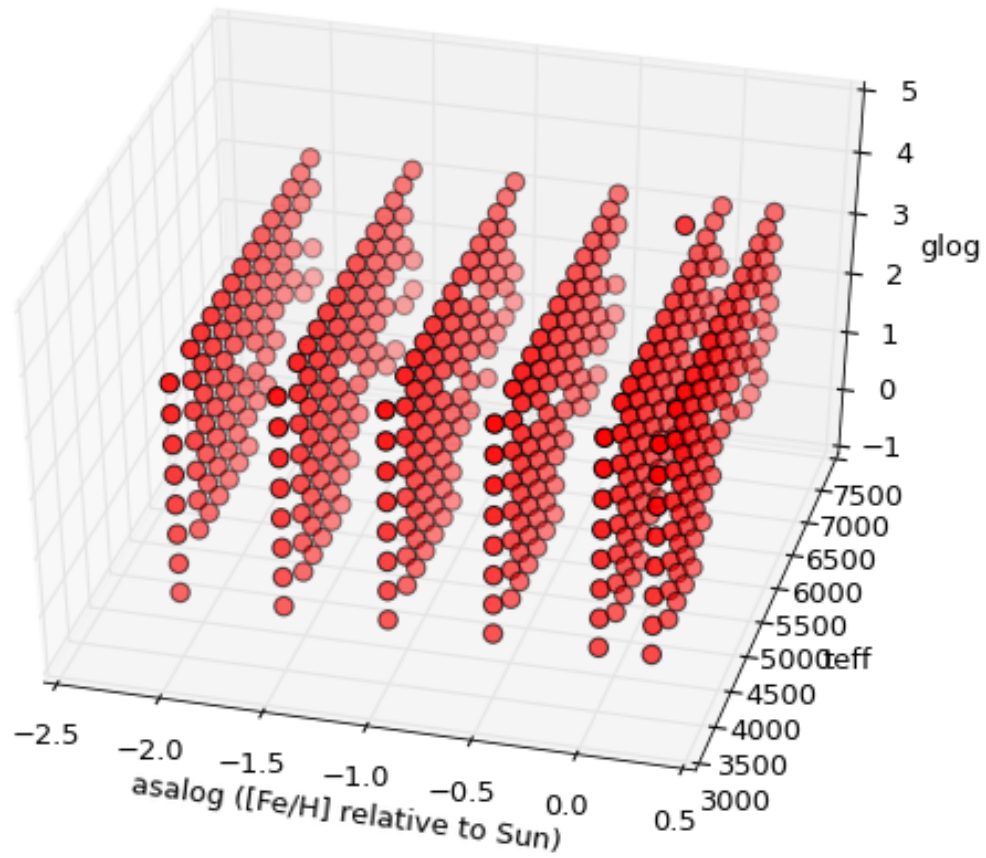
Figure 5.2: – 3D grid of atmospheric models. The scatterplot in the figure shows the (teff, glog, [Fe/H]) values for all existing atmospheric models in the grid (this is the file "grid.moo" provided). The uppermost point represents the Sun.
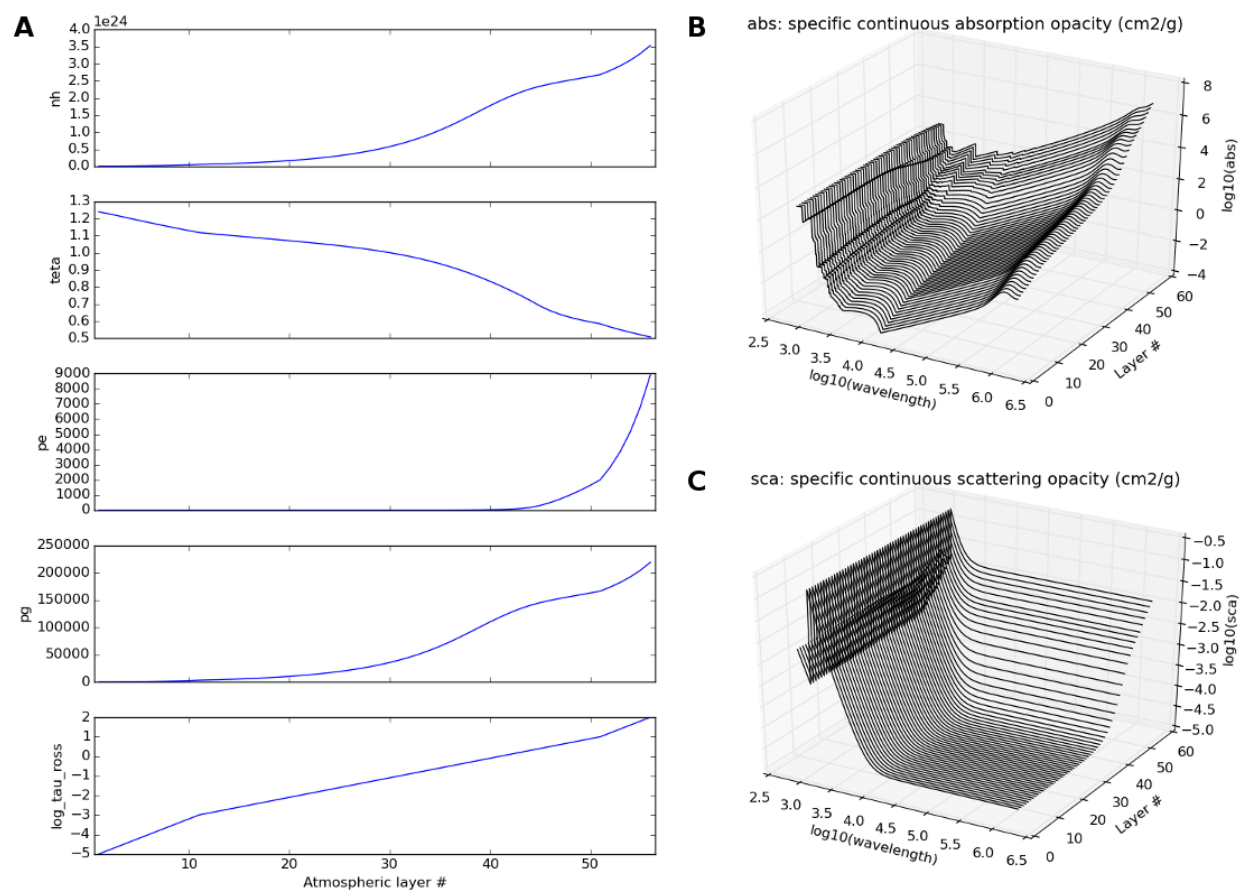
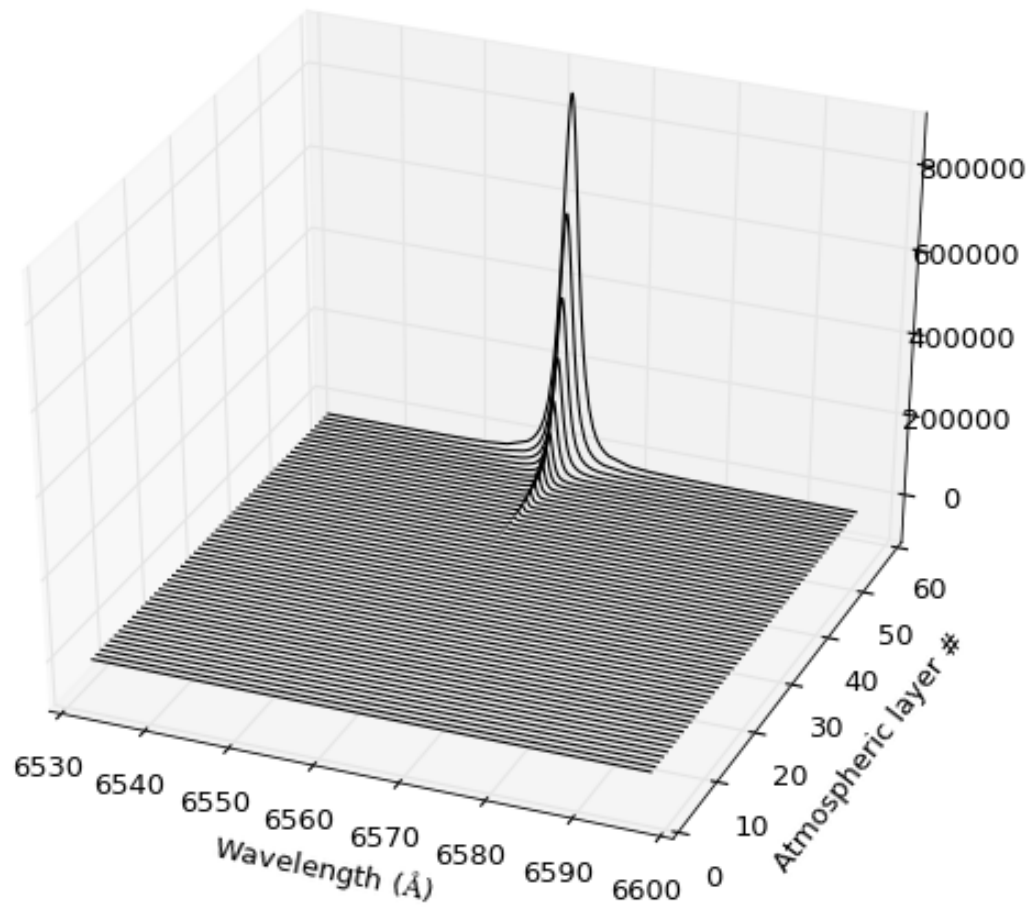Figure 5.3: – Atmospheric model information (Sun). **(A)** data in file modeles.mod; **(B)**, **(C)** data in modeles.opa

Figure 5.4: – Example of H-alpha line profile calculated by `hydro2`.

Files created by `pfant`

Table 5 - Files created by `pfant`

Default name	Description
flux.spec	un-normalized flux (erg/cm**2/s/Hz multiplied by 10**5)
flux.cont	continuum flux (erg/cm**2/s/Hz multiplied by 10**5)
flux.norm	normalized flux (un-normalized flux)/(continuum flux)
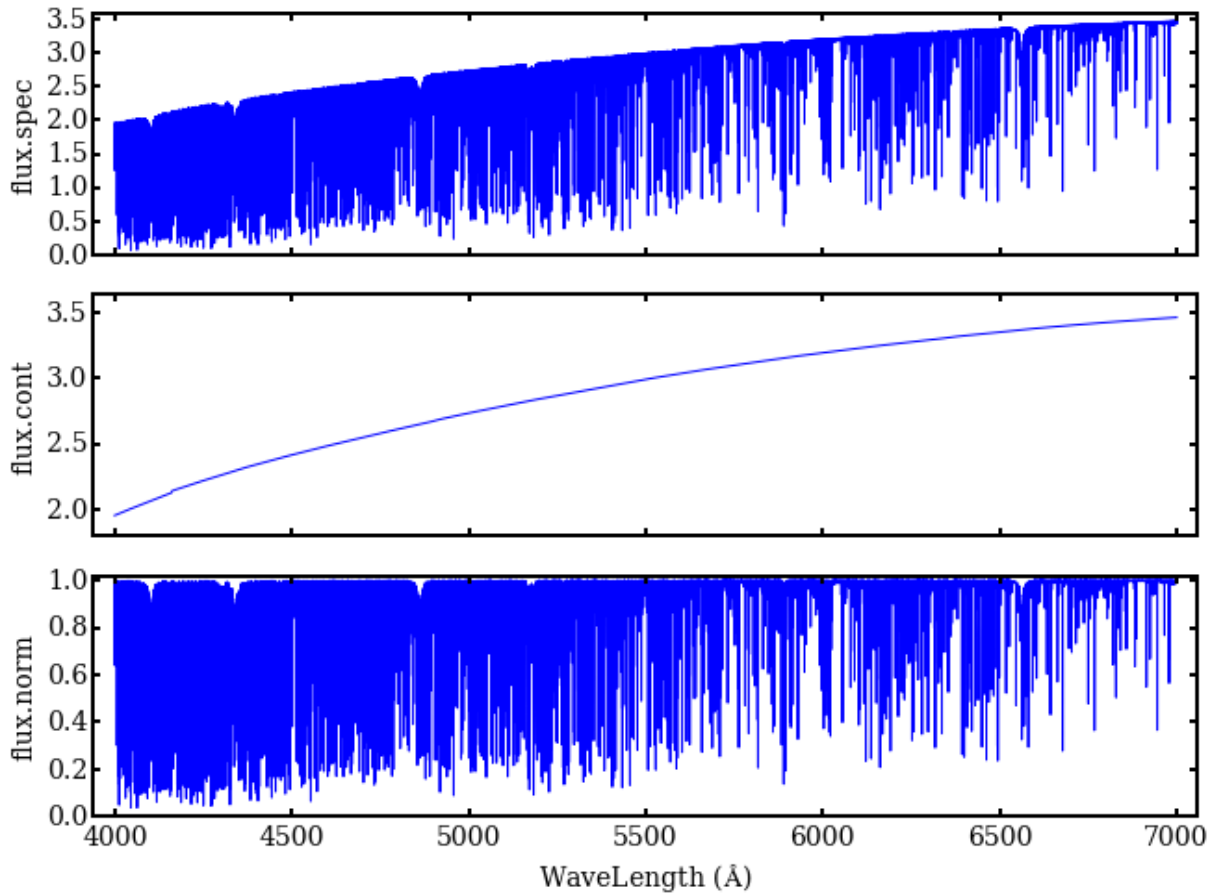
Figure 5.5: – plots showing three `pfant` output files for the [4000, 7000] angstrom region: calculated spectrum; continuum; normalized spectrum. The common prefix "flux" can be changed in file "main.dat" to give a set of files with different names.

Files created by `nulbad`

`nulbad` creates a file whose name by default is the full input file name with the FWHM added with three decimal places. For example,

```
nulbad --fwhm 1.2
```

creates a file named "flux.norm.nulbad.1.200".

To change this, use option "–fn_cv", for example,

```
nulbad --fwhm 1.2 --fn_cv another-name
```

Todo: Another page containing the visual maps of the text files, such as main.dat

CONVERSION OF MOLECULAR LINES LISTS

6.1 Introduction

This section describes the algorithm for conversion of molecular lines lists.

Source formats.

- Robert Kurucz molecular line lists *[Kurucz]* (functional)

- HITRAN Online database *[Gordon2016]* (partially implemented)

- VALD3 *[VALD3]* (to do)

- TurboSpectrum *[Plez]* (to do)

Conversion output:

- PFANT molecular lines file (such as "molecules.dat")

Note: This is work in progress and the conversion is still in experimental development stage.

6.2 How to convert molecular lines

This section is a short tutorial on converting molecular linelists.

1. Create a "project" directory

2. Run `moldbed.py` (Figure 6.1) and press "Ctrl+D" to spawn a new *molecular constants database* in your local directory. The name of such file defaults to "moldb.sqlite". This operation only needs to be carried out once for each "project" directory

3. Download linelist file, e.g., from *[Kurucz]*

4. Run `convmol.py`

5. In the first tab (Figure 6.2)

   - Press "Ctrl+D" (only if the form is disabled)

   - Fill in the form as desired

   - Press "Ctrl+S" to save configuration file for this tab

6. In the second tab (`convmol1`):

   - press "Ctrl+D" (only if the form is disabled)

- select "Kurucz" as data source on the left
- locate linelist file
- select isotope
- most probably, check flags as in (`convmol1`)
- specify output filename, or click on the plant button to make it up
- Press "Ctrl+S" to save configuration file for this tab
- **Click on "Run conversion" button**. Wait for conversion to complete

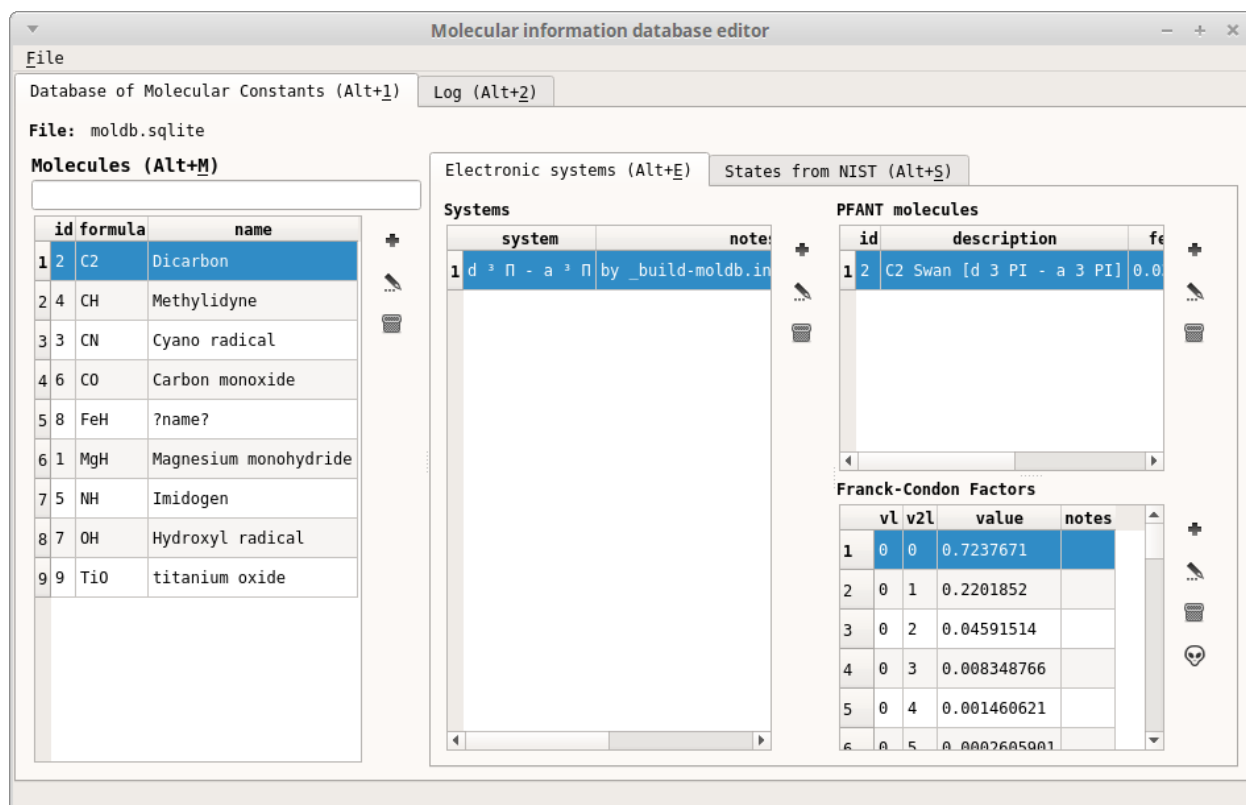7. In the third tab (`convmol2`), see details about the conversion session.

Figure 6.1: – First tab of `convmol.py`

6.3 How the conversion is made

This section describes the conversion algorithm itself.

6.3.1 List of symbols

Input molecular constants obtained from NIST database (all given in unit: cm**-1)

- *omega_e*: vibrational constant – first term
- *omega_ex_e*: vibrational constant – second term
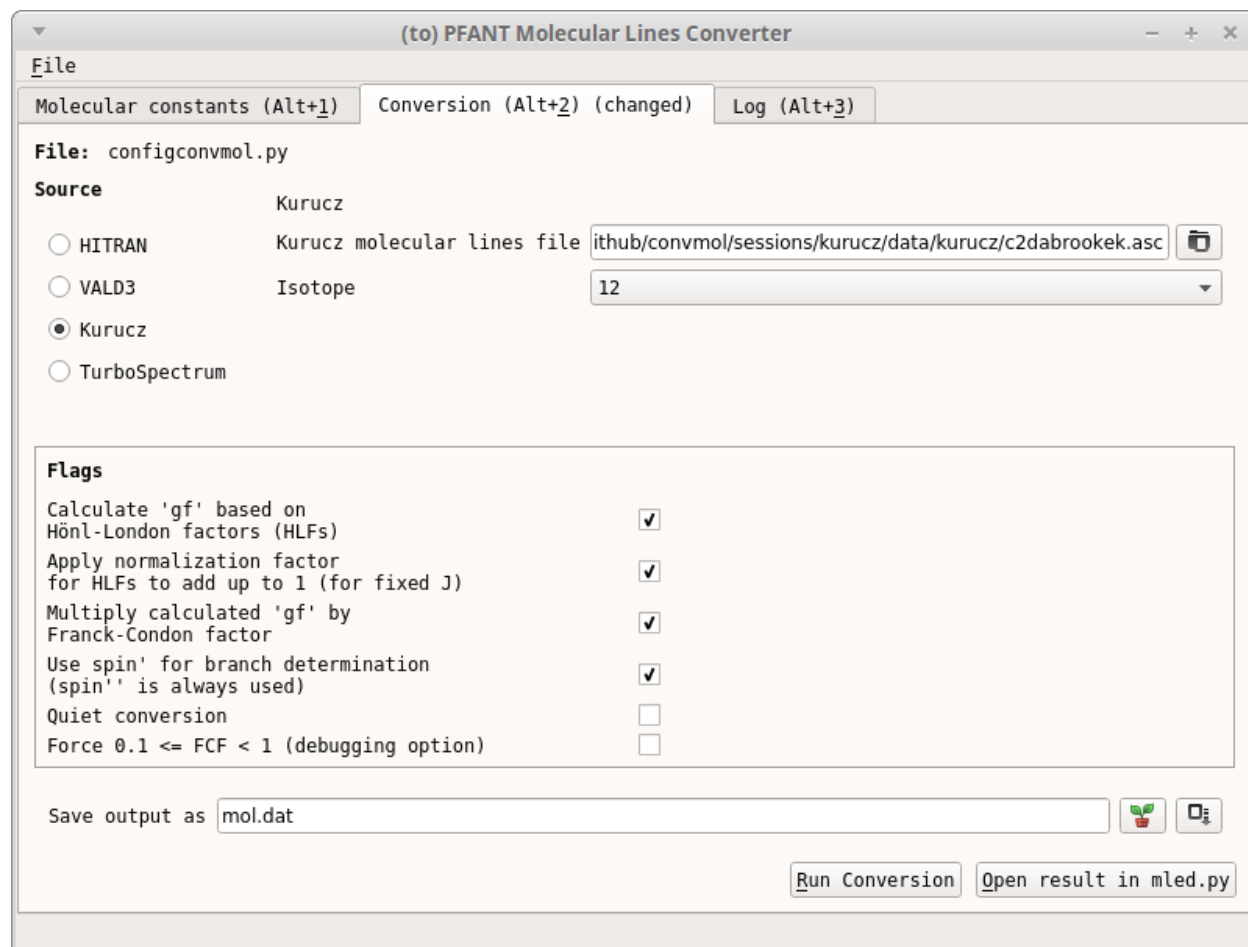
Figure 6.2: – First tab of `convmol.py`
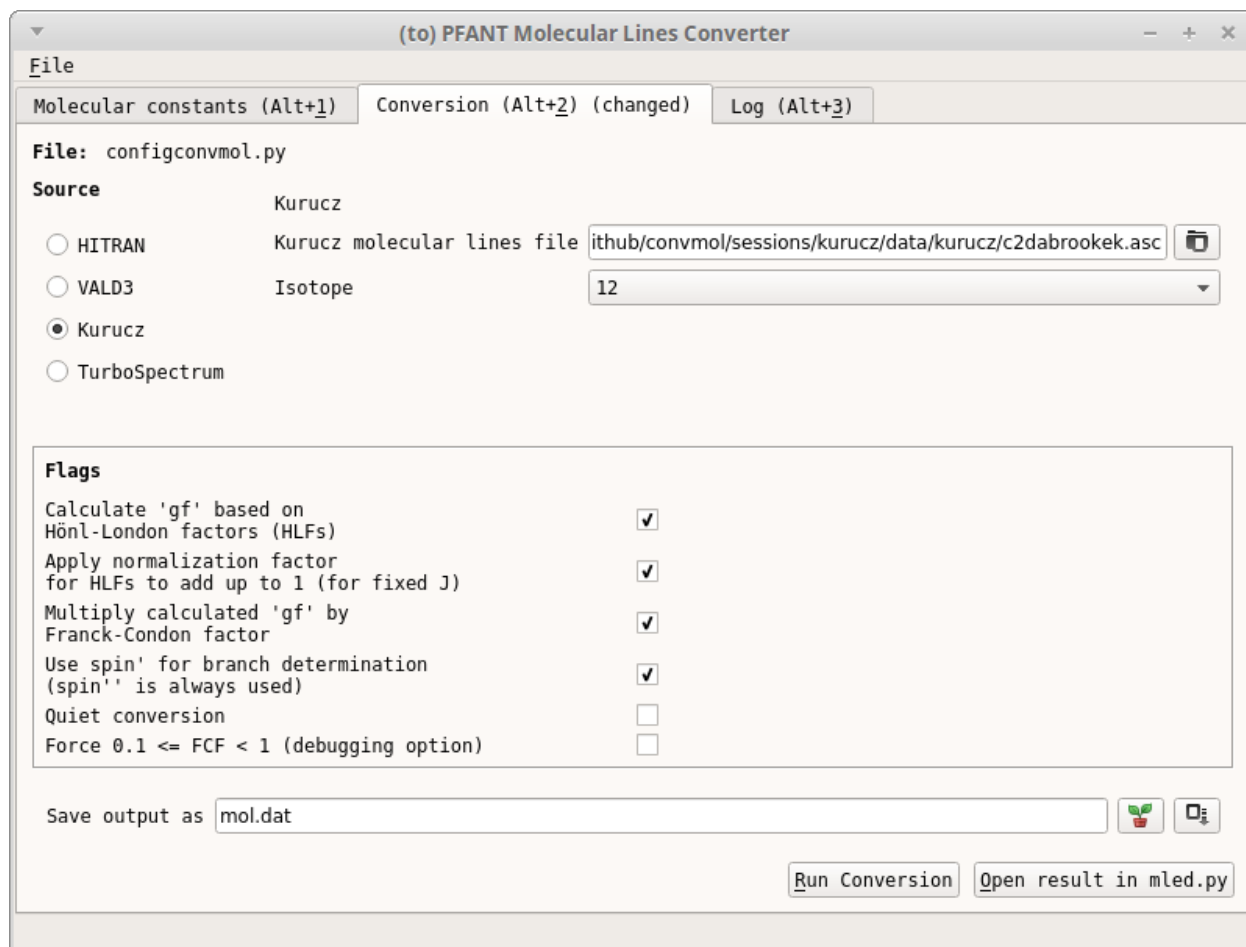
Figure 6.3: – Second tab of `convmol.py`

Figure 6.4: – Third tab of `convmol.py`

- *omega_ey_e*: vibrational constant – third term
- *B_e*: rotational constant in equilibrium position
- *alpha_e*: rotational constant – first term
- *D_e*: centrifugal distortion constant
- *beta_e*: rotational constant – first term, centrifugal force
- *A*: Coupling counstant
- *M2l*: multiplicity of the initial state (1 for singlet, 2 for doublet, 3 for triplet and so on)
- *M2l*: multiplicity of the final state
- *LambdaL*: Sigma/Pi/Delta/Phi of the initial state (0 for Sigma, 1 for Pi, 2 for Delta, 3 for Phi)
- *Lambda2L*: Sigma/Pi/Delta/Phi of the initial state

Input data from line list files

- *lambda*: wavelength (angstrom)
- *vl*: vibrational quantum number of the initial state
- *v2l*: vibrational quantum number of the final state
- *spinl*
- *spin2l*
- *JL*: rotational quantum number of the initial state
- *J2l*: rotational quantum number of the final state

Calculated outputs

The following values are calculated using application `convmol.py` and stored as a PFANT molecular lines file (such as "molecules.dat").

Jl/*J2l*-independent

- *qv*: Franck-Condon factor
- *Bv*: rotational constant
- *Dv*: rotational constant
- *Gv*: rotational constant

These terms are calculated as follows:

```
qv = qv(molecule, system, vl, v2l) is calculated using the TRAPRB code [TRAPRB1970]
                                    The Franck-Condon factors were already calculate␣
→for several
                                    different molecules and are tabulated inside file
→"moldb.sqlite"

Bv = B_e - alpha_e * (v2l + 0.5)

Dv = (D_e + beta_e * (v2l + 0.5)) * 1.0e+06
```

```
Gv = omega_e * (v2l + 0.5) - omega_ex_e * (v2l + 0.5) ** 2 + omega_ey_e * (v2l + 0.5)␣
↪** 3 -
    omega_e / 2.0 - omega_ex_e / 4.0 + omega_ey_e / 8.0
```

Jl/*J2l*-dependent (*i.e.*, for each spectral line)

- *LS*: line strength for given by formulas in *[Kovacs1969]*, Chapter 3; Hönl-London factor
- *S*: normalized line strength

LS is calculated using a different formula depending on:

1. the multiplicities of the transition (currently implemented only cases where the initial and final state have same multiplicity)

2. the value and/or sign of (*DeltaLambda = LambdaL - Lambda2l*);

3. whether *A* is a positive or negative number;

4. the branch of the spectral line (see below how to determine the branch)

So:

```
formula = KovacsFormula(i, ii, iii, iv)

LS = formula(almost every input variable)
```

Todo: Explain term formulas "u+/-", "c+/-"

6.3.2 Normalization of the line strength

Normalization is applied so that, for a given *J2l*,:

```
sum([S[branch] for branch in all_branches]) == 1
```

To achieve this:

```
S = LS * 2. / ((2 * spin2l + 1) * (2 * J2l + 1) * (2 - delta_k))
```

Where:

```
spin2l = (M2l-1)/2
```

6.3.3 How to determine the branch

The branch "label" follows one of the following conventions:

```
singlets: branch consists of a "<letter>", where letter may be either "P", "Q", or "R"

doublets, triplets etc:

    if spin == spinl == spin2l: branch consists of "<letter><spin>"
```

```
    if spinl <> spin2l: branch consists of "<letter><spinl><spin2l>"
```

The branch letter is determined as follows:

```
if Jl < J2l:  "P"
if Jl == J2l: "Q"
if Jl > J2l:  "R"
```

6.4 Where the conversion code is located

- The line strength formulas from *[Kovacs1969]* are in module `pyfant.kovacs` (source code directly available for inspection at https://github.com/trevisanj/pyfant/blob/master/pyfant/kovacs.py)

- The conversion routines are in subpackage `pyfant.convmol` (source code at https://github.com/trevisanj/pyfant/tree/master/pyfant/convmol)

6.5 Bibliography

MISCELLANEA HOW-TO

7.1 Create grid of atmospheric models

For spectral synthesis, PFANT uses model atmospheres encoded in its own binary ".mod" format. Such ".mod" files are generated by the Fortran binary `innewmarcs`, which interpolates within a grid of model atmospheres. This grid is also encoded in ".mod" format and can be generated using the script `create-grid.py`.

To create a grid of atmospheric models:

1. Download atmospheric models of interest from MARCS website (http://marcs.astro.uu.se/) and put all files in a single directory

2. Run one of the options below:

Without MARCS opacities:

```
create-grid.py --mode modbin
```

With MARCS opacities:

> create-grid.py –mode opa

7.2 Converting "VALD3 extended" format atomic lines to PFANT format

The Vienna Atomic Line Database (VALD) is "a collection of atomic and molecular transition parameters of astronomical interest" (http://vald.astro.uu.se/).

To convert from the "VALD3 extended" to a "PFANT atomic lines" file:

```
vald3-to-atoms.py <vald3-extended-filename>
tune-zinf <output-from-previous-command>
```

This is done in two steps. The first step, `vald3-to-atoms.py` does the actual conversion (which is quick) and saves a file, *e.g.*, "atoms-untuned-xxxxx.dat"

The second step (which is time-consuming) is performed by `tune-zinf.py` and aims to tune an important parameter used by the `pfant` Fortran binary.

For more information, see help for `vald3-to-atoms.py`, `tune-zinf.py`, `cut-atoms.py` (call these scripts with `--help` option).

7.3 Continuous opacities: selecting between PFANT and MARCS co-efficients

By default, PFANT internally calculates the continuum absorption coefficients, then adds MARCS scattering coefficients.

PFANT-calculated continuum absorption + MARCS scattering (default)

```
innewmarcs --absoru T --opa T --sca T --abs F
pfant --absoru T --opa T --sca T --abs F
```

or

```
run4.py --opa T --sca T --abs F
```

PFANT-calculated continuum only

```
innewmarcs --absoru T --opa F
pfant --absoru T --opa F
```

or

```
run4.py --absoru T --opa F
```

MARCS opacities (absorption and scattering) only

```
innewmarcs --absoru F --opa T --sca T --abs T
pfant --absoru F --opa T --sca T --abs T
```

or

```
run4.py --absoru F --opa T --sca T --abs T
```

Note: For `innewmarcs`, "–opa T" causes the creation of an additional file *modeles.opa* besides *modeles.mod*.

Related command-line options (also accessible in `x.py`):

```
--opa T ...... switches on MARCS opacities
               (may be of two types: absorption and scattering)
--abs T ...... switches on MARCS absorption
--sca T ...... switches on MARCS scattering
--absoru F ... switches off PFANT internal calculation
```

Note: In order to use continuum opacities calculated by MARCS code (http://marcs.astro.uu.se/), you will need to create your own atmospheric model grid (using `create-grid.py`), or download file "grid.moo" as explained below (this file is too big to be stored on GitHub (241 MB > 100 MB)). File "grid.moo" contains a 3D grid of MARCS (http://marcs.astro.uu.se/) atmospheric models with opacities included.

1. go to directory `PFANT/data/common` and run `get-grid.moo.sh`, or

2. download it from this location (or this location) and save it as "PFANT/data/common/grid.moo"

7.4 Conversion of molecular lines from other formats to PFANT format

See *Conversion of molecular lines lists*.

CHAPTER

EIGHT

TROUBLESHOOTING

8.1 Tracking down why Fortran binaries fail to run

- If you are running the Fortran binaries directly, error messages are printed on screen.

- If you are running from `x.py`, the "Runnables Manager" window has a "Collect errors" button.

- If you are running `run4.py`, open `explorer.py` and click the "Collect errors" button.

`run4.py` or `x.py` create **session directories** named `session-xxx` containing at least these two files:

- *commands.log* – this is the command lines used to invoke the Fortran binaries. This file may be useful if you want to specifically reproduce one of these commands for debugging reasons. In such case, copy-paste this line in your console to see how the Fortran binary runs.

- *fortran.log* – Fortran output is logged into this file.

Your current directory will also have a file named *python.log*, containing debug/info/warning/error messages from Python.

8.2 Metallicity/temperature/gravity of star is outside range in the grid of models

You can activate option `--allow True` to make bypass this check.

This can be done in the command line, *e.g.*, `run4.py --allow T`, or check option "–alow" in Tab 3 of `x.py`.

> **Attention:** Beware that with `--allow True` the interpolation for the atmospheric model may be innapropriate. With that option, `innewmarcs` will **not** extrapolate the atmospheric model grid, but will instead "project" your (teff, glog, metallicity) coordinate onto the closest grid wall, then interpolate.

39

CHEATSHEET

- **Description of stellar parameters, running settings, command-line options:** run `x.py` and navigate through the fields in Tabs 1 and 3. As you navigate, a description of the current field will be displayed at the bottom of the window.

- Call a program with `--help` option, *e.g.*, `pfant --help`.

- `programs.py -p pyfant` lists all Fortran programs in the PFANT pack, and Python programs in the PyFANT project.

9.1 PFANT Fortran binaries

- `innewmarcs`: interpolate atmospheric model

- `hydro2`: calculate hydrogen lines

- `pfant`: spectral synthesis

- `nulbad`: convolution with Gaussian

9.2 Python applications from project PyFANT

9.2.1 Graphical applications

- `abed.py`: Abundances file editor

- `ated.py`: Atomic lines file editor

- `convmol.py`: Conversion of molecular lines data to PFANT format

- `explorer.py`: F311 Explorer – list, visualize, and edit data files (*à la* File Manager)

- `mained.py`: Main configuration file editor

- `mced.py`: Editor for molecular constants file

- `mled.py`: Molecular lines file editor

- `moldbed.py`: Editor for molecules SQLite database

- `tune-zinf.py`: Tunes the "zinf" parameter for each atomic line in atomic lines file

- `x.py`: PFANT Launcher – Graphical Interface for Spectral Synthesis

9.2.2 Command-line tools

- `copy-star.py`: Copies stellar data files (such as main.dat, abonds.dat, dissoc.dat) to local directory

- `create-grid.py`: Merges several atmospheric models into a single file (*i.e.*, the "grid")

- `cut-atoms.py`: Cuts atomic lines file to wavelength interval specified

- `cut-molecules.py`: Cuts molecular lines file to wavelength interval specified

- `cut-spectrum.py`: Cuts spectrum file to wavelength interval specified

- `hitran-scraper.py`: Retrieves molecular lines from the HITRAN database

- `link.py`: Creates symbolic links to PFANT data files as an alternative to copying these (sometimes large) files into local directory

- `nist-scraper.py`: Retrieves molecular constants from NIST Web Book for a particular molecule

- `plot-spectra.py`: Plots spectra on screen or creates PDF file

- `run4.py`: Runs the four Fortran binaries in sequence: `innewmarcs`, `hydro2`, `pfant`, `nulbad`

- `save-pdf.py`: Looks for files "*.norm" inside directories session-* and saves one figure per page in a PDF file

- `vald3-to-atoms.py`: Converts VALD3 atomic/molecular lines file to PFANT atomic lines file.

9.3 Command-line options for the Fortran binaries

Options are accompained by their default values.

Below, `<main_xxxxx>` means that option `xxxxx` is, by default, read from PFANT main configuration file.

9.3.1 `hydro2`

```
hydro2 \
    --logging_console T \
    --logging_dump F \
    --logging_fn_dump <executable name>_dump.log \
    --fn_main main.dat \
    --fn_modeles modeles.mod \
    --fn_absoru2 absoru2.dat \
    --fn_hmap hmap.dat \
    --interp 1 \
    --kik 0 \
    --ptdisk <main_llzero>  \
    --llfin <main_llfin>  \
    --amores T \
    --kq 1 \
    --zph 12.00
```

9.3.2 `innewmarcs`

```
innewmarcs \
    --logging_console T \
    --logging_dump F \
    --logging_fn_dump <executable name>_dump.log \
```

```
    --fn_main main.dat \
    --fn_modeles modeles.mod \
    --fn_modgrid grid.mod \
    --fn_moo grid.moo \
    --allow F \
    --fn_opa modeles.opa \
    --opa T
```

9.3.3 `pfant`

```
pfant \
    --logging_console T \
    --logging_dump F \
    --logging_fn_dump <executable name>_dump.log \
    --fn_main main.dat \
    --fn_modeles modeles.mod \
    --fn_absoru2 absoru2.dat \
    --fn_hmap hmap.dat \
    --interp 1 \
    --kik 0 \
    --ptdisk <main_llzero>  \
    --llfin <main_llfin>  \
    --fn_opa modeles.opa \
    --fn_partit partit.dat \
    --fn_abonds abonds.dat \
    --fn_atoms atoms.dat \
    --no_molecules F \
    --no_atoms F \
    --no_h F \
    --pas <main_pas>  \
    --aint <main_aint> \
    --opa T \
    --abs F \
    --opa T \
    --opa T \
    --fn_dissoc dissoc.dat \
    --fn_molecules molecules.dat \
    --flprefix <main_flprefix>
```

9.3.4 `nulbad`

```
nulbad \
    --logging_console T \
    --logging_dump F \
    --logging_fn_dump <executable name>_dump.log \
    --fn_main main.dat \
    --flprefix <main_flprefix>  \
    --fn_flux <main_flprefix>.norm  \
    --flam F \
    --fn_cv <flux file name>.nulbad.<fwhm> \
    --pat <main_pas>  \
    --convol T \
    --fwhm <main_fwhm>
```

PFANT on GitHub: http://github.com/trevisanj/PFANT

PyFANT on GitHub: http://github.com/trevisanj/pyfant

BIBLIOGRAPHY

[Kovacs1969] Istvan Kovacs, Rotational Structure in the spectra of diatomic molecules. American Elsevier, 1969

[TRAPRB1970] Jarmain, W. R., and J. C. McCallum. "TRAPRB: a computer program for molecular transitions." University of Western Ontario (1970)

[NIST] http://webbook.nist.gov/chemistry/

[Kurucz] http://kurucz.harvard.edu/molecules.html

[VALD3] http://vald.astro.univie.ac.at/~vald3/php/vald.php

[Plez] http://www.pages-perso-bertrand-plez.univ-montp2.fr/

[Gordon2016] I.E. Gordon, L.S. Rothman, C. Hill, R.V. Kochanov, Y. Tan, P.F. Bernath, M. Birk, V. Boudon, A. Campargue, K.V. Chance, B.J. Drouin, J.-M. Flaud, R.R. Gamache, J.T. Hodges, D. Jacquemart, V.I. Perevalov, A. Perrin, K.P. Shine, M.-A.H. Smith, J. Tennyson, G.C. Toon, H. Tran, V.G. Tyuterev, A. Barbe, A.G. Császár, V.M. Devi, T. Furtenbacher, J.J. Harrison, J.-M. Hartmann, A. Jolly, T.J. Johnson, T. Karman, I. Kleiner, A.A. Kyuberis, J. Loos, O.M. Lyulin, S.T. Massie, S.N. Mikhailenko, N. Moazzen-Ahmadi, H.S.P. Müller, O.V. Naumenko, A.V. Nikitin, O.L. Polyansky, M. Rey, M. Rotger, S.W. Sharpe, K. Sung, E. Starikova, S.A. Tashkun, J. Vander Auwera, G. Wagner, J. Wilzewski, P. Wcisło, S. Yu, E.J. Zak, The HITRAN2016 Molecular Spectroscopic Database, J. Quant. Spectrosc. Radiat. Transf. (2017). doi:10.1016/j.jqsrt.2017.06.038.