F311 Documentation

Release 17.9.27.0

Julio Trevisan

Nov 29, 2017

CONTENTS

Welcome!

CHAPTER

ONE

INTRODUCTION

`f311` is a Python 3 package containing many resources on selected topics in Astronomy. Once installed, the package makes available a collection of scripts and an API (application programming interface).

1.1 Using the API

The API is organized in sub-packages, which can be imported as follows:

```
import f311.pyfant as pf
import f311.convmol as cm
import f311.explorer as ex
import f311.filetypes as ft
import f311.physics as ph
import f311.aosss as ao
```

For convenience, the symbols of the subpackages are exposed at the root package level, so another way to import the API is:

```
import f311
```

1.2 Contributing to this project

If you would like to contribute to this project, you can clone the source code on GitHub.

1.3 List of scripts

1.3.1 f311.aosss – âĂIJAdaptive Optics Systems Simulation SupportâĂİ

Graphical applications

- *wavelength-chart.py*: Draws chart showing spectral lines of interest, spectrograph wavelength ranges, ESO atmospheric .

Command-line tools

- *create-simulation-reports.py*: Creates HTML reports from WebSim-COMPASS output files
- *create-spectrum-lists.py*: Create several .splist (spectrum list) files from WebSim-COMPASS output files; groups r
- *get-compass.py*: Downloads WebSim-COMPASS simulations
- *list-mosaic-modes.py*: Lists MOSAIC Spectrograph modes
- *organize-directory.py*: Organizes simulation directory (creates folders, moves files, creates âĂŸindex.htmlâĂŹ)

1.3.2 f311.convmol – Conversion of molecular lines files.

Graphical applications

- *convmol.py*: Conversion of molecular lines data to PFANT format
- *mced.py*: Editor for molecular constants file
- *moldbed.py*: Editor for molecules SQLite database

Command-line tools

- *hitran-scraper.py*: Retrieves molecular lines from the HITRAN database [Gordon2016]
- *nist-scraper.py*: Retrieves and prints a table of molecular constants from the NIST Chemistry Web Book.

1.3.3 f311.explorer – Object-oriented framework to handle file types:

Graphical applications

- *abed.py*: Abundances file editor
- *ated.py*: Atomic lines file editor
- *cubeed.py*: Data Cube Editor, import/export WebSim-COMPASS data cubes
- *explorer.py*: F311 Explorer – list, visualize, and edit data files (_Ãă **la**_ File Manager)
- *mained.py*: Main configuration file editor.
- *mled.py*: Molecular lines file editor.
- *optionsed.py*: PFANT command-line options file editor.
- *splisted.py*: Spectrum List Editor
- *tune-zinf.py*: Tunes the âĂIJzinfâĂİ parameter for each atomic line in atomic lines file

Command-line tools

- *create-grid.py*: Merges several atmospheric models into a single file (_i.e._, the âĂIJgridâĂİ)
- *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
- *plot-spectra.py*: Plots spectra on screen or creates PDF file
- *vald3-to-atoms.py*: Converts VALD3 atomic/molecular lines file to PFANT atomic lines file.

1.3.4 f311.pyfant – Python interface to the PFANT spectral synthesis software (Fortran)

Graphical applications

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

Command-line tools

- *copy-star.py*: Copies stellar data files (such as main.dat, abonds.dat, dissoc.dat) to local directory
- *link.py*: Creates symbolic links to PFANT data files as an alternative to copying these (sometimes large) files into local directoy
- *merge-molecules.py*: Merges several PFANT molecular lines file into a single one
- run-multi.py‘: Runs pfant and nulbad in âĂIJmulti modeâĂİ (equivalent to Tab 4 in `x.py <autoscripts/script-run-multi.py`: Runs pfant and nulbad in âĂIJmulti mod)
- *run4.py*: Runs the four Fortran binaries in sequence: *innewmarcs*, *hydro2*, *pfant*, *nulbad*

Hint: You can use `programs.py` to list available scripts.

1.4 List of acronyms

API – application programming interface

GUI – graphical user interface

FWHM – full with at half maximum

1.5 Acknowledgement

The project started in 2015 at IAG-USP (Institute of Astronomy, Geophysics and Atmospheric Sciences at University of SÃčo Paulo, Brazil).

Partially funded by FAPESP - Research Support Foundation of the State of SÃčo Paulo, Brazil (2015-2017).

CHAPTER

TWO

F311 INSTALLATION

Python 3 version required: Python 3.4.6+, Python 3.5.3+, or Python 3.6+ (https://packaging.python.org/guides/migrating-to-pypi-org/)

2.1 Method 1: Using Anaconda without virtual environment

This will make AnacondaâĂŹs Python 3 the default `python` command for your user account. Make sure you donâĂŹt mind this, otherwise follow Method 2.

First install Anaconda or Miniconda. When you do so, please make sure that you **answer âĂIJyesâĂİ to this (or similar) question**:

```
Do you wish the installer to prepend the Miniconda3 install location
to PATH in your /home/j/.bashrc ? [yes|no]
>> yes
```

After Anaconda/Miniconda installation, close the terminal and open it again so that your PATH is updated. **Or if your shell is bash**, you can just type `source ~/.bashrc` on the terminal.

Now install some packages using pip:

```
pip install numpy scipy matplotlib astropy configobj bs4 robobrowser requests␣
↪fortranformat tabulate rows pyqt5 a99 f311
```

2.2 Method 2: Using Anaconda virtual environment

This method uses a **conda** virtual environment. It works with a separate installation of Python and related packages.

First you will need to have Anaconda or Miniconda installed. If you have none of these installed yet, just install Miniconda.

Once Anaconda/Miniconda is installed, create a new virtual environment called âĂIJastroenvâĂİ (or any name you like):

```
conda create --name astroenv python=3.5 # or 3.6
```

Activate this new virtual environment:

```
source activate astroenv
```

Now install the packages:

```
pip install numpy scipy matplotlib astropy configobj bs4 lxml robobrowser requests␣
↪fortranformat tabulate rows pyqt5 a99 f311
```

Note Every time you want to work with F311, you will need to activate the environment:

```
source activate astroenv
```

To deactivate the environment:

```
source deactivate
```

2.3 Method 3: Developer mode

This allows you to pull the most recent version of the code directly from GitHub.

First, follow Method 1 or 2 above, **but do not install f311**, *i.e.*, the pip command should be:

```
pip install numpy scipy matplotlib astropy configobj bs4 lxml robobrowser requests␣
↪fortranformat tabulate rows pyqt5 a99
```

Second, clone the f311 GitHub repository:

```
git clone ssh://git@github.com/trevisanj/f311.git
```

or

```
git clone http://github.com/trevisanj/f311
```

Finally, install F311 in **developer** mode:

```
cd f311
python setup.py develop
```

2.4 Troubleshooting installation

2.4.1 MatPlotLib and PyQt5

```
ValueError: Unrecognized backend string "qt5agg": valid strings are ['GTKAgg',
↪'template', 'pdf',
'GTK3Agg', 'cairo', 'TkAgg', 'pgf', 'MacOSX', 'GTK', 'WX', 'GTKCairo', 'Qt4Agg', 'svg
↪', 'agg',
'ps', 'emf', 'WebAgg', 'gdk', 'WXAgg', 'CocoaAgg', 'GTK3Cairo']
```

Solution: update Matplotlib to version 1.4 or later

2.4.2 Problems with package bs4

```
bs4.FeatureNotFound: Couldn't find a tree builder with the features you requested:␣
↪lxml. Do you need to install a parser library?
```

Solution: install package âĂIJlxmlâĂİ: `pip install lxml`

CHAPTER

THREE

PACKAGE F311

The root package provides the collaboration model implementation.

The collaboration model allows `f311.COLLABORATORS` to contribute with

- `DataFile` subclasses, i.e., implement handling (load, save, etc.) of new file types;
- `Vis` subclasses, i.e., implement visualizations for these files;
- Standalone scripts placed in the directory `packagename/scripts`

3.1 Print file handling classes information

```
"""Lists different subsets of DataFile subclasses"""
import f311

titles = ("text", "binary", "1D spectrum")
allclasses = (f311.classes_txt(), f311.classes_bin(), f311.classes_sp())

for title, classes in zip(titles, allclasses):
    print("\n*** Classes that can handle {} files***".format(title))
    for cls in classes:
        print("{:25}: {}".format(cls.__name__, cls.__doc__.strip().split("\n")[0]))
```

The output should be something like:

```
*** Classes that can handle text files***
FileAbXFwhm               : `x.py` Differential Abundances and FWHMs (Python source)
FileAbonds                : PFANT Stellar Chemical Abundances
FileAbsoru2               : "Absoru2" file
FileAtoms                 : PFANT Atomic Lines
FileConfigConvMol         : Python source containing 'config_conv = ConfigConv(...)
FileDissoc                : PFANT Stellar Dissociation Equilibrium Information
FileFCF                   : File containing Franck-Condon Factors (FCFs)
FileHmap                  : PFANT Hygrogen Lines Map
FileKuruczMolecule        : Kurucz molecular lines file
FileKuruczMoleculeBase    : Base class for the two types of Kurucz molecular lines file
FileKuruczMoleculeOld     : Kurucz molecular lines file, old format #0
FileKuruczMoleculeOld1    : Kurucz molecular lines file, old format #1
FileMain                  : PFANT Stellar Main Configuration
FileModTxt                : MARCS Atmospheric Model (text file)
FileMolConsts             : Python source containing 'fobj = MolConsts(...)
FileMolecules             : PFANT Molecular Lines
FileOpa                   : MARCS ".opa" (opacity model) file format.
```

```
FileOptions              : `x.py` Command-line Options
FilePar                  : WebSim-COMPASS ".par" (parameters) file
FilePartit               : PFANT Partition Function
FilePlezTiO              : Plez molecular lines file, TiO format
FilePy                   : Configuration file saved as a .py Python source script
FilePyConfig             : Base class for config files. Inherit and set class␣
↪variable 'modulevarname' besides usual
FileSpectrum             : Base class for all files representing a single 1D spectrum
FileSpectrumNulbad       : PFANT Spectrum (`nulbad` output)
FileSpectrumPfant        : PFANT Spectrum (`pfant` output)
FileSpectrumXY           : "Lambda-flux" Spectrum (2-column text file)
FileToH                  : PFANT Hydrogen Line Profile
FileVald3                : VALD3 atomic or molecular lines file

*** Classes that can handle binary files***
FileFullCube             : FITS Data Cube ("full" opposed to "sparse")
FileGalfit               : FITS file with frames named INPUT_*, MODEL_*, RESIDUAL_*,␣
↪which is the output of Galfit software
FileHitranDB             : HITRAN Molecules Catalogue
FileModBin               : PFANT Atmospheric Model (binary file)
FileMolDB                : Database of Molecular Constants
FileMoo                  : Atmospheric model or grid of models (with opacities␣
↪included)
FileSQLiteDB             : Represents a SQLite database file.
FileSparseCube           : FITS Sparse Data Cube (storage to take less disk space)
FileSpectrumFits         : FITS Spectrum
FileSpectrumList         : FITS Spectrum List

*** Classes that can handle 1D spectrum files***
FileSpectrum             : Base class for all files representing a single 1D spectrum
FileSpectrumFits         : FITS Spectrum
FileSpectrumNulbad       : PFANT Spectrum (`nulbad` output)
FileSpectrumPfant        : PFANT Spectrum (`pfant` output)
FileSpectrumXY           : "Lambda-flux" Spectrum (2-column text file)
```

3.1.1 API reference

autodoc/f311

CHAPTER

FOUR

SPECTRAL SYNTHESIS

Welcome!!

pyfant is a Python interface for the PFANT Spectral Synthesis Software for Astronomy.

Spectral synthesis softwares have a fundamental role in Astronomy. It is a crucial step in determining stellar properties - such as temperature, metallicity, and chemical abundances - in which the synthetic spectrum (or a combination of several of these) is compared with the measured spectrum of a star or a whole stellar population either by the full spectrum fitting, spectral energy distribution or specific spectral lines and regions. It is of great interest that the software has a comprehensive and intuitive user interface and easiness of parameter input and its multiple variations, and also tools for incorporating data like atomic/molecular lines, atmospheric models, etc.

Package `f311.pyfant` provides a Python interface to the PFANT Fortran binaries, including the ability to run the Fortran binaries in parallel in a multi-processing scheme via API or GUI.

manipulate and save PFANT data files using `f311.filetypes`, allow for complex batch operations.

4.1 Applications

The applications related to package f311.pyfant are listed below. For them to work, you need to install PFANT.

The PFANT Quick Start serves as a guide to using some of these applications.

4.1.1 Graphical applications

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

4.1.2 Command-line tools

- `copy-star.py` – Copies stellar data files (such as main.dat, abonds.dat, dissoc.dat) to local directory
- `link.py` – Creates symbolic links to PFANT data files as an alternative to copying these (sometimes large) files into local directory
- `run4.py` – Runs the four Fortran binaries in sequence: `innewmarcs`, `hydro2`, `pfant`, `nulbad`
- `save-pdf.py` – Looks for files âĂIJ*.normâĂİ inside directories session-* and saves one figure per page in a PDF file

4.2 Coding using the API

This section contains a series of examples on how to use the PFANT Fortran executables from a Python script. These âĂIJbindingsâĂİ to the Fortran binaries, together with the ability to load, manipulate and save PFANT data files using `f311.filetypes`, allow for complex batch operations.

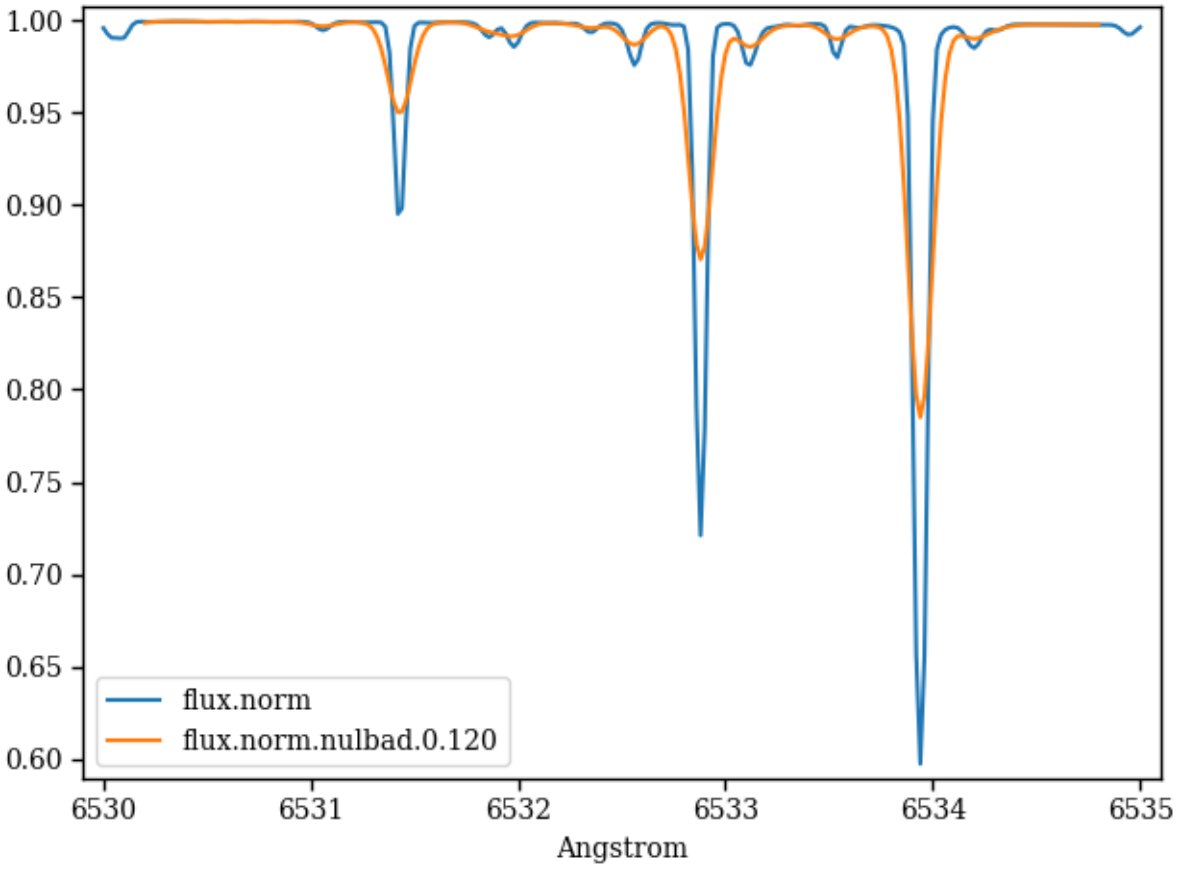
4.2.1 Spectral synthesis

```
"""Runs synthesis over short wavelength range, then plots normalized and convolved␣
↪spectrum"""

import f311.pyfant as pf
import f311.explorer as ex
import matplotlib.pyplot as plt


if __name__ == "__main__":
    # Copies files main.dat and abonds.dat to local directory (for given star)
    pf.copy_star(starname="sun-grevesse-1996")
    # Creates symbolic links to all non-star-specific files, such as atomic &␣
↪molecular lines,
    # partition functions, etc.
    pf.link_to_data()

    # # First run
    # Creates object that will run the four Fortran executables (innewmarcs, hydro2,␣
↪pfant, nulbad)
    obj = pf.Combo()
    # synthesis interval start (angstrom)
    obj.conf.opt.llzero = 6530
    # synthesis interval end (angstrom)
    obj.conf.opt.llfin = 6535

    # Runs Fortrans and hangs until done
    obj.run()

    # Loads result files into memory. obj.result is a dictionary containing elements .
↪..
    obj.load_result()
    print("obj.result = {}".format(obj.result))
    res = obj.result
    plt.figure()
    ex.draw_spectra_overlapped([res["norm"], res["convolved"]])
    plt.savefig("norm-convolved.png")
    plt.show()
```

4.2.2 Spectral synthesis - convolutions

The following example convolves the synthetic spectrum (file âĂIJflux.normâĂİ) with Gaussian profiles of different FWHMs.

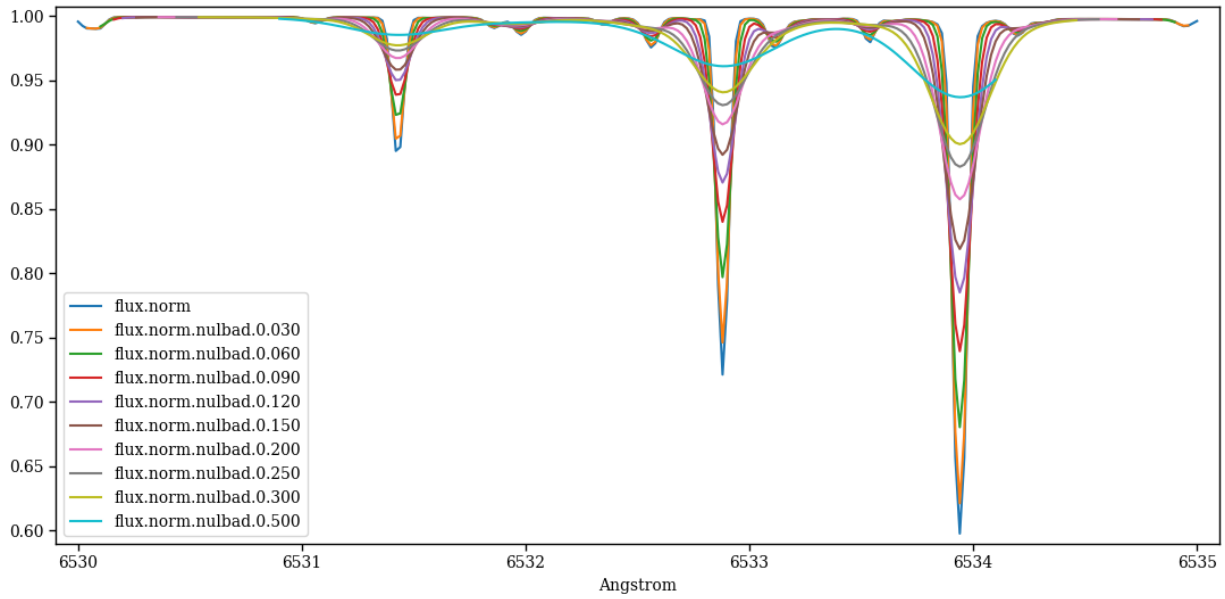
```
#!/usr/bin/env python

"""Runs synthesis over short wavelength range, then plots normalized and convolved␣
↪spectrum"""
```


1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
flux.norm
flux.norm.nulbad.0.120
6530
6531
6532
6533
6534
6535
Angstrom

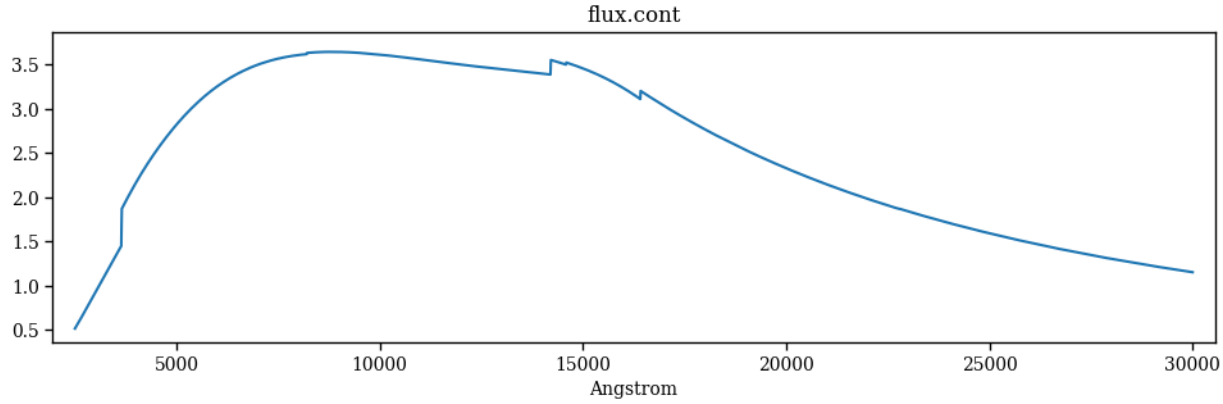
```
import f311.pyfant as pf
import f311.explorer as ex
import matplotlib.pyplot as plt
import a99

# FWHM (full width at half of maximum) of Gaussian profiles in angstrom
FWHMS = [0.03, 0.06, 0.09, 0.12, 0.15, 0.20, 0.25, 0.3, 0.5]

if __name__ == "__main__":
    # Copies files main.dat and abonds.dat to local directory (for given star)
    pf.copy_star(starname="sun-grevesse-1996")
    # Creates symbolic links to all non-star-specific files
    pf.link_to_data()

    # # 1) Spectral synthesis
    # Creates object that will run the four Fortran executables (innewmarcs, hydro2,␣
↪pfant, nulbad)
    ecombo = pf.Combo()
    # synthesis interval start (angstrom)
    ecombo.conf.opt.llzero = 6530
    # synthesis interval end (angstrom)
    ecombo.conf.opt.llfin = 6535
    # Runs Fortrans and hangs until done
    ecombo.run()
    ecombo.load_result()
    # Retains un-convolved spectrum for comparison
    spectra = [ecombo.result["norm"]]

    # # 2) Convolutions
    for fwhm in FWHMS:
        enulbad = pf.Nulbad()
        enulbad.conf.opt.fwhm = fwhm
        enulbad.run()
        enulbad.load_result()
        # Appends convolved spectrum for comparison
        spectra.append(enulbad.result["convolved"])

    # # 3) Plots
    plt.figure()
    ex.draw_spectra_overlapped(spectra)
    K = 1.1
    a99.set_figure_size(plt.gcf(), 1000*K, 500*K)
    plt.tight_layout()
    plt.savefig("many-convs.png")
    plt.show()
```

4.2.3 Spectral synthesis - Continuum

```
"""Runs synthesis over large wavelength range, then plots continuum"""

import f311.pyfant as pf
import f311.explorer as ex
import matplotlib.pyplot as plt
import a99
```




```
if __name__ == "__main__":
    # Copies files main.dat and abonds.dat to local directory (for given star)
    pf.copy_star(starname="sun-grevesse-1996")
    # Creates symbolic links to all non-star-specific files, such as atomic &␣
↪molecular lines,
    # partition functions, etc.
    pf.link_to_data()

    # Creates object that will run the four Fortran executables (innewmarcs, hydro2,␣
↪pfant, nulbad)
    obj = pf.Combo()
    oo = obj.conf.opt
    # synthesis interval start (angstrom)
    oo.llzero = 2500
    # synthesis interval end (angstrom)
    oo.llfin = 30000
    # savelength step (angstrom)
    oo.pas = 1.
    # Turns off hydrogen lines
    oo.no_h = True
    # Turns off atomic lines
    oo.no_atoms = True
    # Turns off molecular lines
    oo.no_molecules = True

    obj.run()
    obj.load_result()
    print("obj.result = {}".format(obj.result))
    res = obj.result
    ex.draw_spectra([res["cont"]], setup=ex.PlotSpectrumSetup(fmt_ylabel=None))
    K = .75
    a99.set_figure_size(plt.gcf(), 1300*K, 450*K)
    plt.tight_layout()
    plt.savefig("continuum.png")
    plt.show()
```



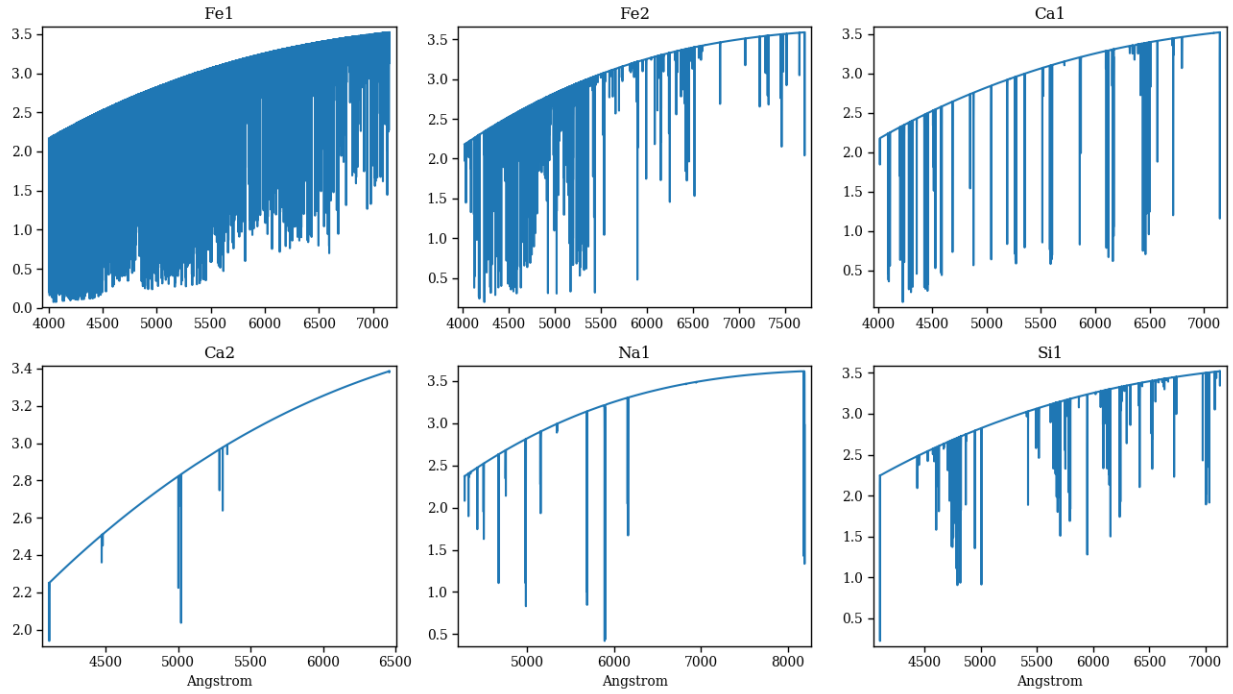
4.2.4 Spectral synthesis - Separate atomic species

PFANT atomic lines files contains wavelength, log_gf and other tabulated information for several (element, ionization level) atomic species.

The following code calculates isolated atomic spectra for a list of arbitrarily chosen atomic species.

```
"""Runs synthesis for specified atomic species separately. No molecules or hydrogen␣
↪lines."""

import f311.pyfant as pf
import f311.explorer as ex
import matplotlib.pyplot as plt
import f311.filetypes as ft
import a99

# ["<element name><ionization level>", ...]
MY_SPECIES = ["Fe1", "Fe2", "Ca1", "Ca2", "Na1", "Si1"]

if __name__ == "__main__":
    pf.copy_star(starname="sun-grevesse-1996")
    pf.link_to_data()

    # Loads full atomic lines file
    fatoms = ft.FileAtoms()
    fatoms.load()


    runnables = []
    for elem_ioni in MY_SPECIES:
        atom = fatoms.find_atom(elem_ioni)

        # Creates atomic lines file object containing only one atom
        fatoms2 = ft.FileAtoms()
        fatoms2.atoms = [atom]

        ecombo = pf.Combo()
        # Overrides file "atoms.dat" with in-memory object
        ecombo.conf.file_atoms = fatoms2
```

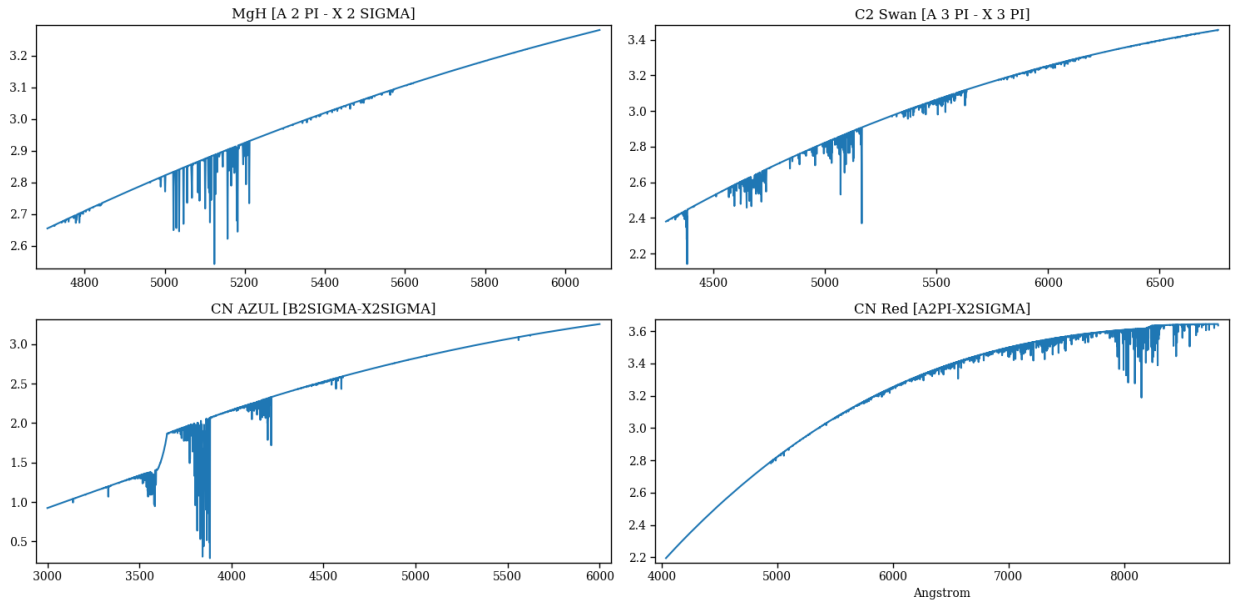
```
        ecombo.conf.flag_output_to_dir = True
        oo = ecombo.conf.opt
        # Assigns synthesis range to match atomic lines range
        oo.llzero, oo.llfin = fatoms2.llzero, fatoms2.llfin
        # Turns off hydrogen lines
        oo.no_h = True
        # Turns off molecular lines
        oo.no_molecules = True

        runnables.append(ecombo)

    pf.run_parallel(runnables)

    # Draws figure
    f = plt.figure()
    a99.format_BLB()
    for i, (title, ecombo) in enumerate(zip(MY_SPECIES, runnables)):
        ecombo.load_result()
        plt.subplot(2, 3, i+1)
        ex.draw_spectra_overlapped([ecombo.result["spec"]],
                                   setup=ex.PlotSpectrumSetup(flag_xlabel=i/3 >= 1,␣
↪flag_legend=False))
        plt.title(title)

    K = 1.
    a99.set_figure_size(plt.gcf(), 1300*K, 740*K)
    plt.tight_layout()
    plt.savefig("synthesis-atoms.png")
    plt.show()
```



4.2.5 Spectral synthesis - Separate molecules

```
"""Runs synthesis for molecular species separately. No atomic nor hydrogen lines."""

import f311.pyfant as pf
import f311.explorer as ex
import matplotlib.pyplot as plt
import f311.filetypes as ft
import a99

SUBPLOT_NUM_ROWS = 2
SUBPLOT_NUM_COLS = 2

if __name__ == "__main__":
    pf.copy_star(starname="sun-grevesse-1996")
    pf.link_to_data()

    # Loads full molecular lines file
    fmol = ft.FileMolecules()
    fmol.load()


    runnables = []
    for molecule in fmol:
        fmol2 = ft.FileMolecules()
        fmol2.molecules = [molecule]

        ecombo = pf.Combo()
        # Overrides file "molecules.dat" with in-memory object
        ecombo.conf.file_molecules = fmol2
        ecombo.conf.flag_output_to_dir = True
        oo = ecombo.conf.opt
        # Assigns synthesis range to match atomic lines range
        oo.llzero, oo.llfin = fmol2.llzero, fmol2.llfin
        # Turns off hydrogen lines
        oo.no_h = True
        # Turns off atomic lines
        oo.no_atoms = True
        # Adjusts the wavelength step according to the calculation interval
        oo.pas = max(1, round(oo.llfin*1./20000/2.5)*2.5)
        oo.aint = max(50., oo.pas)

        runnables.append(ecombo)

    pf.run_parallel(runnables)

    num_panels = SUBPLOT_NUM_COLS*SUBPLOT_NUM_ROWS
    num_molecules = len(runnables)
    ifigure = 0
    a99.format_BLB()
    for i in range(num_molecules+1):
        not_first = i > 0
        first_panel_of_figure = (i / num_panels - int(i / num_panels)) < 0.01
        is_panel = i < num_molecules

        if not_first and (not is_panel or first_panel_of_figure):
            plt.tight_layout()
            K = 1.
```

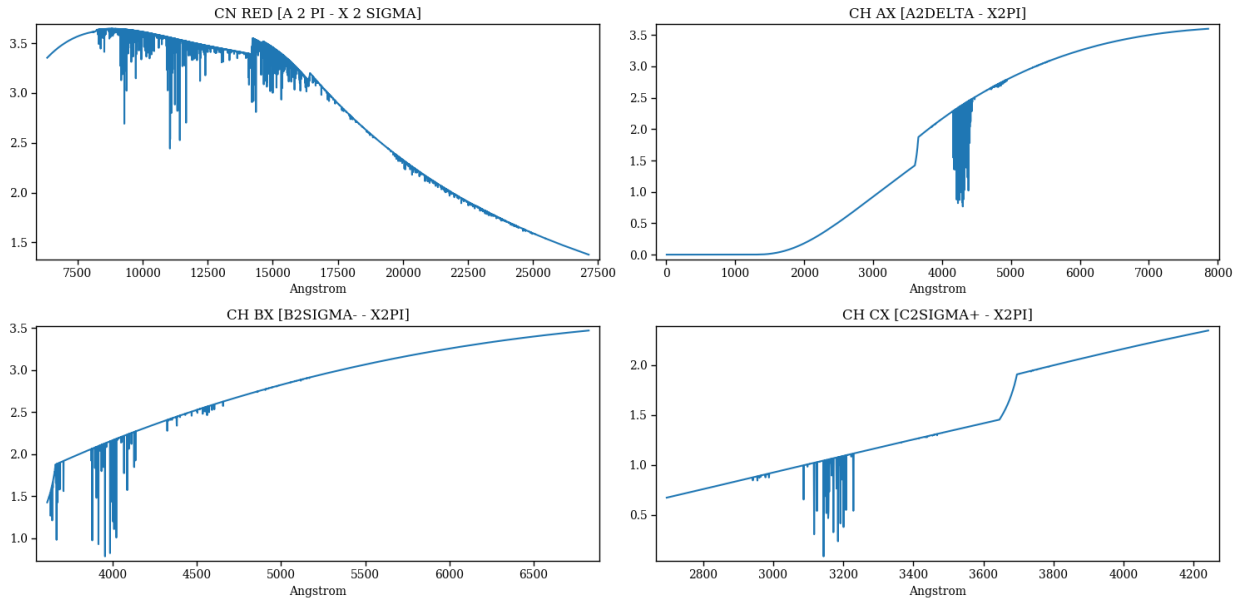
```
            a99.set_figure_size(plt.gcf(), 1500 * K, 740 * K)
            plt.tight_layout()
            filename_fig ="synthesis-molecules-{}.png".format(ifigure)
            print("Saving figure '{}'...".format(filename_fig))
            plt.savefig(filename_fig)
            plt.close()
            ifigure += 1

        if first_panel_of_figure and is_panel:
            plt.figure()

        if is_panel:
            ecombo = runnables[i]
            ecombo.load_result()

            isubplot = i % num_panels + 1
            plt.subplot(SUBPLOT_NUM_ROWS, SUBPLOT_NUM_COLS, isubplot)
            ex.draw_spectra_overlapped([ecombo.result["spec"]],
               setup=ex.PlotSpectrumSetup(flag_xlabel=i/3 >= 1, flag_legend=False))

            _title = fmol[i].description
            if "]" in _title:
                title = _title[:_title.index("]")+1]
            else:
                title = _title[:20]
            plt.title(title)
```



etc.

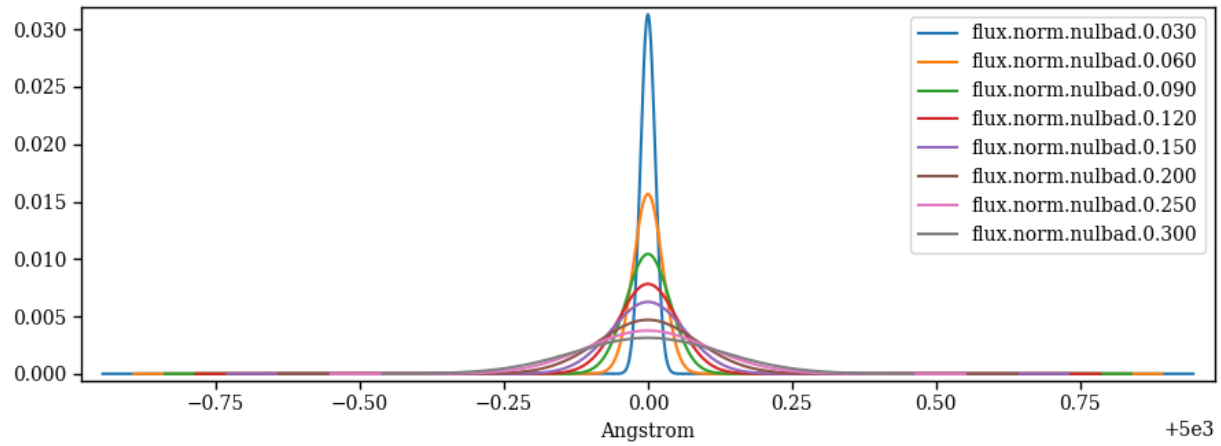
4.2.6 Gaussian profiles as nulbad outputs

`nulbad` is one of the Fortran executables of the PFANT package. It is the one that convolves the synthetic spectrum calculated by `pfant` with a Gaussian profile specified by a âĂIJfwhmâĂİ parameter.




```
"""
Nulbad's "impulse response"

Saves "impulse" spectrum (just a spike at lambda=5000 angstrom) as "flux.norm",
then runs `nulbad` repeatedly to get a range of Gaussian profiles.

"""

import f311.pyfant as pf
import f311.explorer as ex
import matplotlib.pyplot as plt
import a99
import f311.filetypes as ft
import numpy as np


# FWHM (full width at half of maximum) of Gaussian profiles in angstrom
FWHMS = [0.03, 0.06, 0.09, 0.12, 0.15, 0.20, 0.25, 0.3]

if __name__ == "__main__":
    # Copies files main.dat and abonds.dat to local directory (for given star)
    pf.copy_star(starname="sun-grevesse-1996")
    # Creates symbolic links to all non-star-specific files
    pf.link_to_data()

    # # 1) Creates "impulse" spectrum
    fsp = ft.FileSpectrumPfant()
    sp = ft.Spectrum()
    N = 2001
    sp.x = (np.arange(0, N, dtype=float)-(N-1)/2)*0.001+5000
    sp.y = np.zeros((N,), dtype=float)
    sp.y[int((N-1)/2)] = 1.

    fsp.spectrum = sp
    fsp.save_as("flux.norm")
```

```
    # # 2) Convolutions
    spectra = []
    for fwhm in FWHMS:
        enulbad = pf.Nulbad()
        enulbad.conf.opt.fwhm = fwhm
        enulbad.run()
        enulbad.load_result()
        enulbad.clean()
        # Appends convolved spectrum for comparison
        spectra.append(enulbad.result["convolved"])

    # # 3) Plots
    f = plt.figure()
    ex.draw_spectra_overlapped(spectra)
    K = 0.7
    a99.set_figure_size(plt.gcf(), 1300*K, 500*K)
    plt.tight_layout()
    plt.savefig("gaussian-profiles.png")
    plt.show()
```

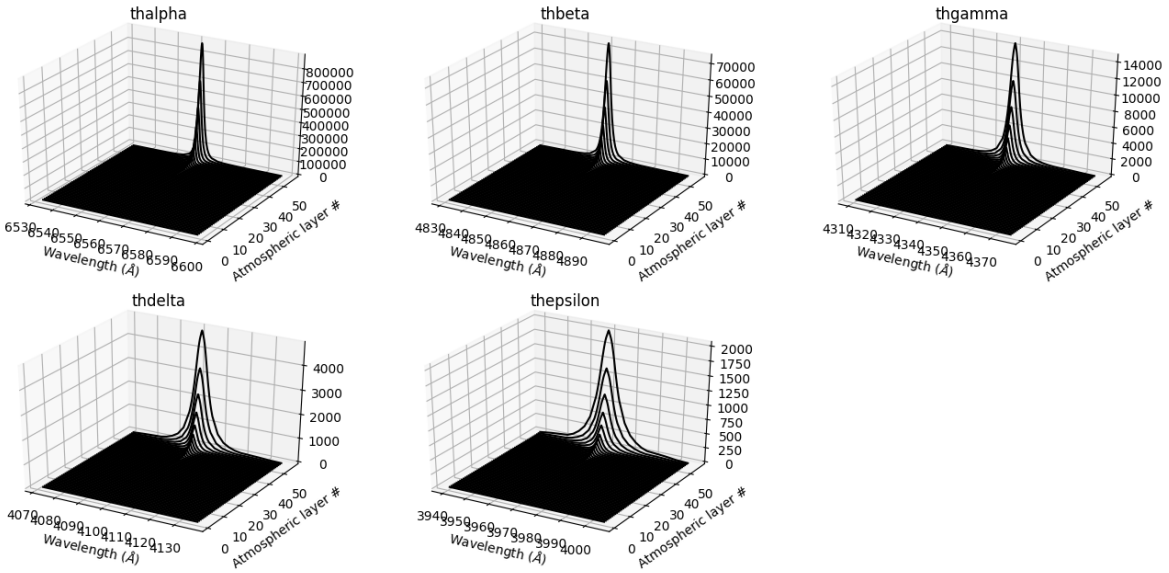


4.2.7 Plot hydrogen profiles

```
"""
Calculates hydrogen lines profiles, then plots them in several 3D subplots
"""

import f311.pyfant as pf
import f311.explorer as ex
import f311.filetypes as ft
import f311.physics as ph
import a99
import os
import shutil
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # yes, required (see below)

def mylog(*args):
```

```
    print("^^ {}".format(", ".join(args)))


def main(flag_cleanup=True):
    tmpdir = a99.new_filename("hydrogen-profiles")

    # Saves current directory
    pwd = os.getcwd()
    mylog("Creating directory '{}'...".format(tmpdir))
    os.mkdir(tmpdir)
    try:
        pf.link_to_data()
        _main()
    finally:
        # Restores current directory
        os.chdir(pwd)
        # Removes temporary directory
        if flag_cleanup:
            mylog("Removing directory '{}'...".format(tmpdir))
            shutil.rmtree(tmpdir)
        else:
            mylog("Not cleaning up.")


def _main():
    fm = ft.FileMain()
    fm.init_default()
    fm.llzero, fm.llfin = 1000., 200000.  # spectral synthesis range in Angstrom

    ei = pf.Innewmarcs()
    ei.conf.file_main = fm
    ei.run()
    ei.clean()

    eh = pf.Hydro2()
    eh.conf.file_main = fm
    eh.run()
    eh.load_result()
    eh.clean()

    _plot_profiles(eh.result["profiles"])


def _plot_profiles(profiles):
    fig = plt.figure()
    i = 0
    for filename, ftoh in profiles.items():
        if ftoh is not None:
            mylog("Drawing '{}'...".format(filename))
            # ax = plt.subplot(2, 3, i+1)
            ax = fig.add_subplot(2, 3, i+1, projection='3d')
            ax.set_title(filename)
            ex.draw_toh(ftoh, ax)
            i += 1

    plt.tight_layout()
    plt.savefig("hydrogen-profiles.png")
    plt.show()
```

```
if __name__ == "__main__":
    main(flag_cleanup=True)
```



4.3 API reference

autodoc/f311.pyfant

CHAPTER

FIVE

CONVERSION OF MOLECULAR LINES LISTS

5.1 Introduction

Conversion between different formats of files containing molecular spectral lines data.

Conversion inputs:

- Robert Kurucz molecular line lists (fully implemented (old and new Kurucz format)) [Kurucz]
- HITRAN Online database (partially implemented)
- VALD3 (to do)
- TurboSpectrum (to do)

Note: This package is currently under construction (2017-11-13)

Conversion output:

- PFANT molecular lines file (such as âĂIJmolecules.datâĂİ)

5.2 Most relevant applications in F311 package

5.2.1 Graphical applications

- *convmol.py*: Conversion of molecular lines data to PFANT format
- *mced.py*: Editor for molecular constants file
- *moldbed.py*: Editor for molecules SQLite database

5.2.2 Command-line tools

- *hitran-scraper.py*: Retrieves molecular lines from the HITRAN database [Gordon2016]
- *nist-scraper.py*: Retrieves and prints a table of molecular constants from the NIST Chemistry Web Book.

5.3 How the conversion is made

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

- *omega_e*: vibrational constant – first term
- *omega_ex_e*: vibrational constant – second term
- *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*: ?SPDF? of the initial state (0 for Sigma, 1 for Pi, 2 for Delta, 3 for Phi)
- *Lambda2L*: ?SPDF? of the initial state

Hint: These values were downloaded from NIST for several molecules and can be navigated through in the applications `convmol.py` or `mced.py`.

Molecular constants can be downloaded from NIST using script `nist-scraper.py`

5.3.2 Input data from line list files (*e.g.* [Kurucz])

- *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

5.3.3 Calculated outputs

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

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 code by Singh [Sing1998].
                                   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; HÃűnl-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)
```

Hint: All the line strength formulas and logic to determine which formula to use are in module `f311.physics.multiplicity`. The latter contains references to the formulas and tables from [Kovacs] that were used for each specific (i, ii, iii, iv) case.

Todo: Explain term formulas âĂIJu+/-âĂIJ, âĂIJc+/-âĂIJ

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
```

How to determine the branch

The branch âĂIJlabelâĂİ 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"
```

5.4 API documentation

autodoc/f311.convmol

5.5 Bibliography

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

[Sing1998] unpublished

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

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

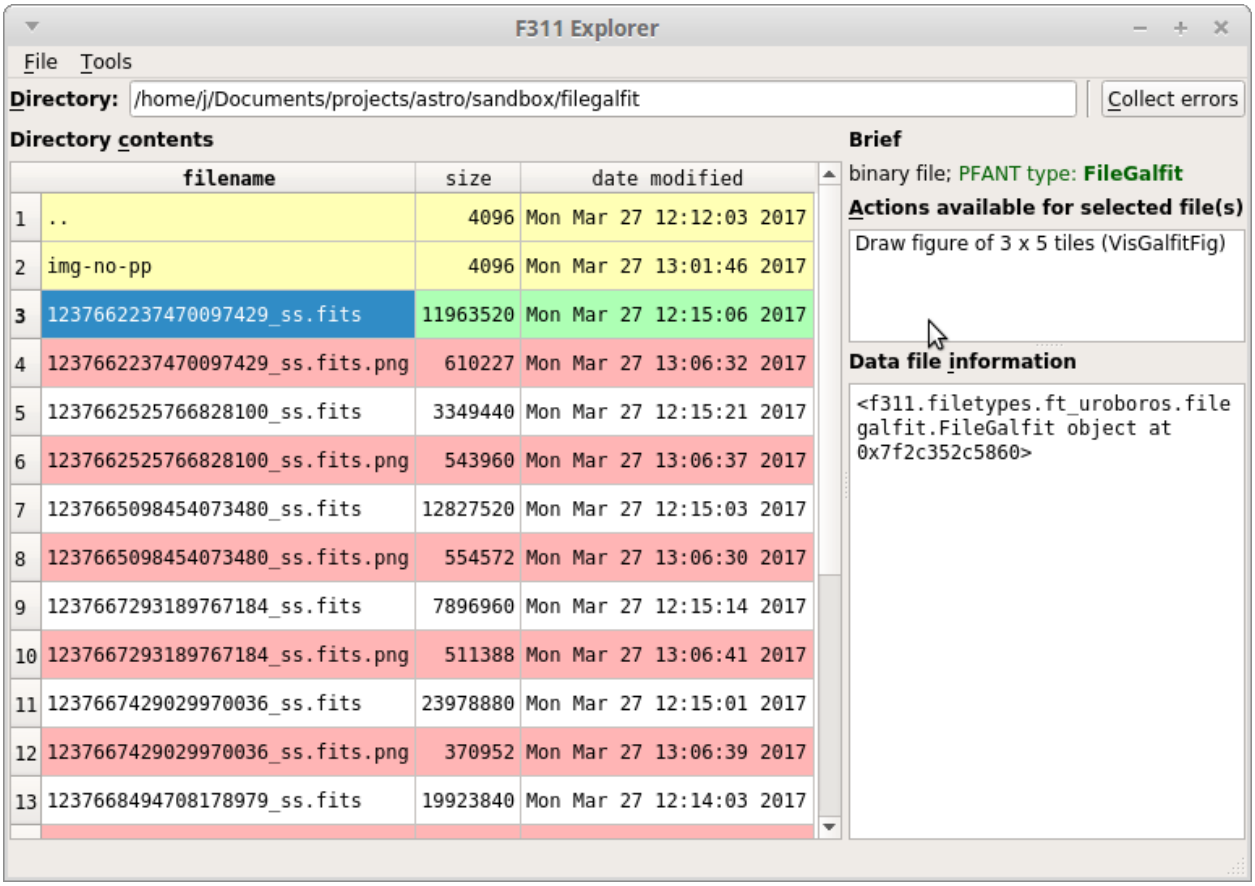
CHAPTER

SIX

FILE EXPLORER AND EDITORS

6.1 Introduction

File edit & visualization, including file-explorer-like `explorer.py` (Figure 6.1).



6.2 List of applications

```
programs.py -p explorer
```

Graphical applications:

- `abed.py` – Abundances file editor
- `ated.py` – Atomic lines file editor
- `cubeed.py` – Data Cube Editor, import/export WebSim-COMPASS data cubes
- `explorer.py` – F311 Explorer – list, visualize, and edit data files (*Ãă la* File Manager)
- `mained.py` – Main configuration file editor.
- `mled.py` – Molecular lines file editor.
- `splisted.py` – Spectrum List Editor
- `tune-zinf.py` – Tunes the âĂIJzinfâĂİ parameter for each atomic line in atomic lines file

Command-line tools:

- `create-grid.py` – Merges several atmospheric models into a single file (_i.e._, the âĂIJgridâĂİ)
- `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
- `plot-spectra.py` – Plots spectra on screen or creates PDF file
- `vald3-to-atoms.py` – Converts VALD3 atomic/molecular lines file to PFANT atomic lines file.

6.3 API reference

autodoc/f311.explorer

CHAPTER

SEVEN

FILE HANDLING API

7.1 Introduction

f311.filetypes represents most of the non-visual essence of the F311 project. the. That package has classes to handle many different file formats used in Astronomy.

All classes descend from `DataFile` containing some basic methods:

- `load()`: loads file from disk into internal object variables
- `save_as()`: saves file to disk
- `init_default()`: initializes file object with default information

7.2 Supported file types

The following table was generated in 12/Nov/2017. The âĂIJEditorâĂİ column shows the applications in the F311 project that can handle these files (*i.e.*, load/edit/save).

All file types are also recognized by `explorer.py`.

Description	Default filename	Class name
âĂIJLambda-fluxâĂİ Spectrum (2-column text file)		FileSpectrumXY
Atmospheric model or grid of models (with opacities included)	grid.moo	FileMoo
Configuration file for molecular lines conversion GUI (Python code)	configconvmol.py	FileConfigConv
Database of Molecular Constants	moldb.sqlite	FileMolDB
FITS Sparse Data Cube (storage to take less disk space)	default.sparsecube	FileSparseCube
FITS Spectrum		FileSpectrumFit
FITS Spectrum List	default.splist	FileSpectrumLis
FITS WebSim Compass Data Cube	default.fullcube	FileFullCube
FITS file with frames named INPUT_*, MODEL_*, RESIDUAL_* (Galfit software output)		FileGalfit
File containing Franck-Condon Factors (FCFs)		FileFCF
HITRAN Molecules Catalogue	hitrandb.sqlite	FileHitranDB
Kurucz molecular lines file		FileKuruczMole
Kurucz molecular lines file, old format #0		FileKuruczMole
Kurucz molecular lines file, old format #1		FileKuruczMole
MARCS âĂIJ.opaâĂİ (opacity model) file format.	modeles.opa	FileOpa
MARCS Atmospheric Model (text file)		FileModTxt
Molecular constants config file (Python code)	configmolconsts.py	FileMolConsts
PFANT âĂIJAbsoru2âĂİ file	absoru2.dat	FileAbsoru2

Table 7.1 – continued from previous page

Description	Default filename	Class name
PFANT Atmospheric Model (binary file)	modeles.mod	FileModBin
PFANT Atomic Lines	atoms.dat	FileAtoms
PFANT Command-line Options	options.py	FileOptions
PFANT Hydrogen Line Profile	thalpha	FileToH
PFANT Hygrogen Lines Map	hmap.dat	FileHmap
PFANT Main Stellar Configuration	main.dat	FileMain
PFANT Molecular Lines	molecules.dat	FileMolecules
PFANT Partition Function	partit.dat	FilePartit
PFANT Spectrum (*nulbad* output)		FileSpectrumNu
PFANT Spectrum (*pfant* output)	flux.norm	FileSpectrumPfa
PFANT Stellar Chemical Abundances	abonds.dat	FileAbonds
PFANT Stellar Dissociation Equilibrium Information	dissoc.dat	FileDissoc
Plez molecular lines file, TiO format		FilePlezTiO
VALD3 atomic or molecular lines file		FileVald3
WebSim-COMPASS âĂIJ.parâĂİ (parameters) file		FilePar
x.py Differential Abundances X FWHMs (Python source)	abxfwhm.py	FileAbXFwhm

By the way, the above table was generated with the following code:

```
import filetypes as ft
print("\n".join(ft.tabulate_filetypes_rest(55)))
```

7.3 Examples

7.3.1 Convert 1D spectral file to FITS format

```
#!/usr/bin/env python
"""Converts 1D spectral file of any supported type to FITS format.

The new file is saved with name "<original-filename>.fits".

TODO handle non-equally spaced wavelength values
"""

import f311.filetypes as ft
import sys
import logging

if __name__ == "__main__":
    if len(sys.argv) < 2 or any([x.startswith("-") for x in sys.argv[1:]]):
        print(__doc__+"\nUsage:\n\n    convert-to-fits.py filename0 [filename1
↪[filename2 [...]]]\n")
        sys.exit()

    for filename in sys.argv[1:]:
        print("Converting file '{}'...".format(filename))

        try:
            spectrum = ft.load_spectrum(filename)

            if spectrum is None:
```

```
                print("File '{}' not recognized as a 1D spectral file".
↪format(filename))
                continue

            filename_new = filename+".fits"

            fnew = ft.FileSpectrumFits()
            fnew.spectrum = spectrum
            fnew.save_as(filename_new)

            print("Successfully saved '{}'".format(filename_new))
        except:
            logging.exception("Error converting file '{}'".format(filename))
```

7.3.2 Import KuruczâĂŹ molecular linelist file

```
"""
This example loads file "c2dabrookek.asc" and prints a memory representation of its␣
↪first line.

This file can be obtained at http://kurucz.harvard.edu/molecules/c2/. First lines of␣
↪file:

```
  287.7558-14.533 23.0 2354.082 24.0 -37095.578 6063a00e1 3d10e3 12 677 34741.495
  287.7564-14.955 22.0 2282.704 23.0 -37024.124 6063a00f1 3d10f3 12 677 34741.419
  287.7582-14.490 21.0 2214.696 22.0 -36955.900 6063a00e1 3d10e3 12 677 34741.205
  287.7613-15.004 24.0 2428.453 25.0 -37169.280 6063a00f1 3d10f3 12 677 34740.828
  287.7650-14.899 20.0 2149.765 21.0 -36890.147 6063a00f1 3d10f3 12 677 34740.382
```
"""

import f311.filetypes as ft

f = ft.load_any_file("c2dabrookek.asc")

print(repr(f[0]).replace(", ", ",\n              "))
```

```
KuruczMolLine(lambda_=2877.558,
              loggf=-14.533,
              J2l=23.0,
              E2l=2354.082,
              Jl=24.0,
              El=37095.578,
              atomn0=6,
              atomn1=6,
              state2l='a',
              v2l=0,
              lambda_doubling2l='e',
              spin2l=1,
              statel='d',
              vl=10,
              lambda_doublingl='e',
              spinl=3,
              iso=12)
```

7.4 API reference

autodoc/f311.filetypes

CHAPTER

EIGHT

SELECTED TOPICS ON PHYSICS

8.1 Introduction

Selected Physics-related resources:

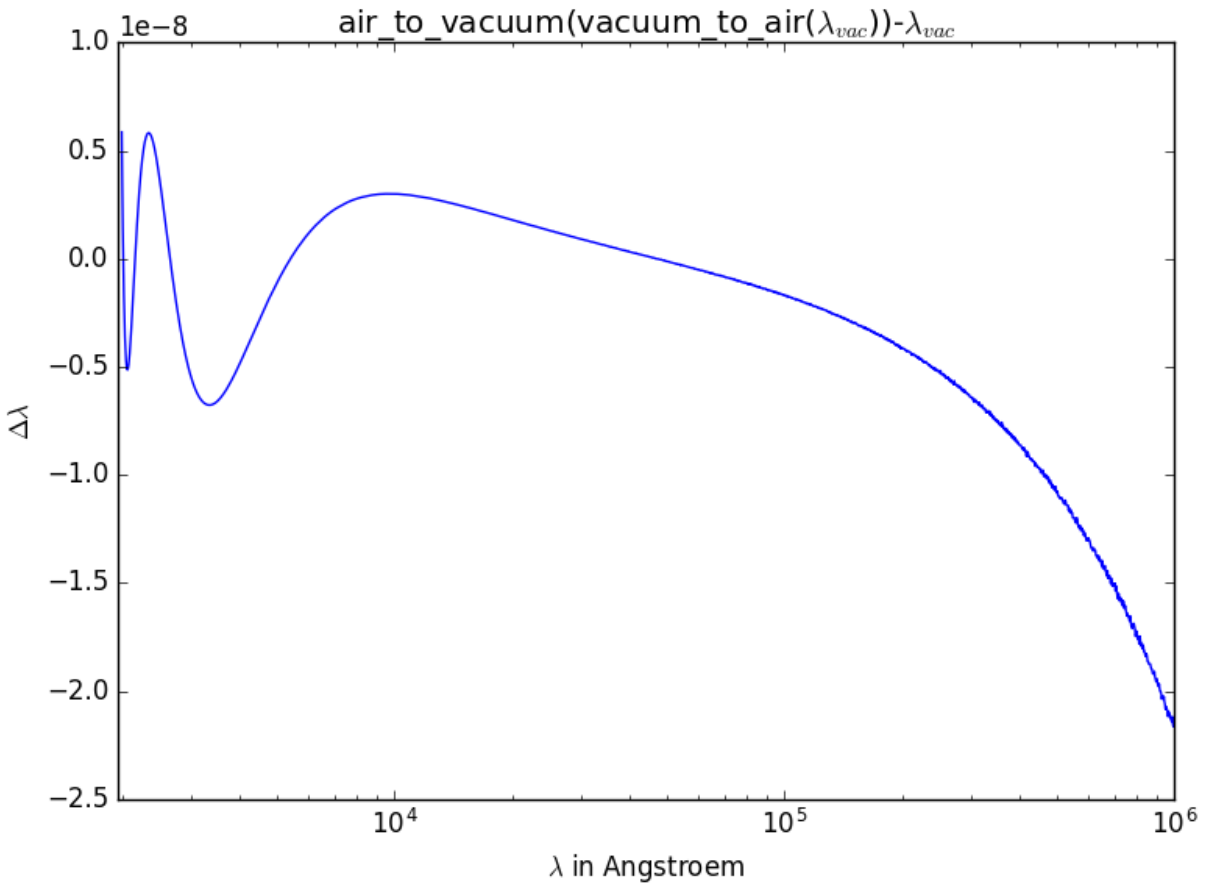
- Photometry (AB/Vega/Standard)
- Spectrum-to-RGB (red, green, blue) color conversion
- Air-to-vacuum (& vice versa) wavelenght conversion
- Calculation of HÃűnl-London factors according to formulas in KovÃącsâĂŹ 1969 [1]

8.2 Examples

8.2.1 Air-to-vacuum (& vice versa) wavelength conversion

The following code reproduces the figure shown in VALD3 Wiki (http://www.astro.uu.se/valdwiki/Air-to-vacuum%20conversion) (âĂIJcomparison of the Morton and the inverse transformation by NP between 2000 ÃĚ and 100000 ÃĚ.âĂİ)

```
import matplotlib.pyplot as plt
import numpy as np
import f311.physics as ph
λvac = 10**np.linspace(np.log10(2000), np.log10(1000000), 2000)
y = ph.air_to_vacuum(ph.vacuum_to_air(λvac))-λvac
plt.semilogx(λvac, y)
plt.xlabel("$\lambda$ in Angstroem")
plt.ylabel("$\Delta\lambda$")
plt.xlim([λvac[0]-50, λvac[-1]])
plt.title("air_to_vacuum(vacuum_to_air($\lambda_{vac}$))-$\lambda_{vac}$")
plt.tight_layout()
plt.show()
```



8.2.2 Calculate the magnitude of a spectrum

The following example compares flux-to-magnitude conversion of the Vega spectrum for different magnitude systems.

```
import f311.physics as ph
import tabulate
systems = ["stdflux", "ab", "vega"]
bands = "UBVRIJHK"
sp = ph.get_vega_spectrum()
rows = [([band]+[ph.calc_mag(sp, band, system) for system in systems]) for band in␣
↪bands]
print(tabulate.tabulate(rows, ["band"]+systems))
```

This code results in the following table:

```
band         stdflux           ab     vega
------  -----------  -----------  ------
U        0.00572505   0.761594        -0
B        0.0696287   -0.10383         -0
V        0.0218067    0.0191189       -0
R        0.0359559    0.214645        -0
I        0.0661095    0.449825        -0
J       -0.0150993    0.874666        -0
H        0.0315447    1.34805         -0
```

```
K         0.0246046    1.85948         -0
```

8.2.3 Calculate HÃűnl-London factors for doublets

In the following examples, a normalization factor is applied to the HÃűnl-London factors (HLF), such that all HLFs for a given J must add up to 1.0:

```
from f311 import physics as ph
S, DELTAK = 0.5, 0  # spin, delta Kronecker
J = 1.5
factor = 2/((2*J+1)*(2*S+1)*(2-DELTAK))
normalized = [f(J)*factor for f in ph.doublet.get_honllondon_formulas(0, 1).values()]
print(sum(normalized))
```

This code should output:

```
1.0
```

The formulas for the HLFs were taken from the book *Istvan Kovacs, âĂIJRotational Structure in the spectra of diatomic molecules. American Elsevier, 1969*

8.3 API reference

autodoc/f311.physics

CHAPTER

NINE

ADAPTIVE OPTICS SYSTEMS SIMULATION SUPPORT (`F311.AOSSS`)

The *aosss* package helps to automatize the simulation of Adaptive Optics Systems.

9.1 Quick Start

9.1.1 List *aosss* applications

```
programs.py -p aosss


Graphical applications
----------------------

  wavelength-chart.py ......... Draws chart showing spectral lines of interest,
                                spectrograph wavelength ranges, ESO atmospheric
                                model, etc.

Command-line tools
------------------

  create-simulation-reports.py  Creates HTML reports from WebSim-COMPASS output
                                files
  create-spectrum-lists.py .... Create several .splist (spectrum list) files
                                from WebSim-COMPASS output files; groups spectra
                                that share same wavelength vector
  get-compass.py ............... Downloads WebSim-COMPASS simulations
  list-mosaic-modes.py ........ Lists MOSAIC Spectrograph modes
  organize-directory.py ....... Organizes simulation directory (creates folders,
                                moves files, creates 'index.html')
```

Note All the programs above can be called with the `--help` or `-h` option for more information

9.1.2 Find wavelength region for simulation

```
wavelength-chart.py
```

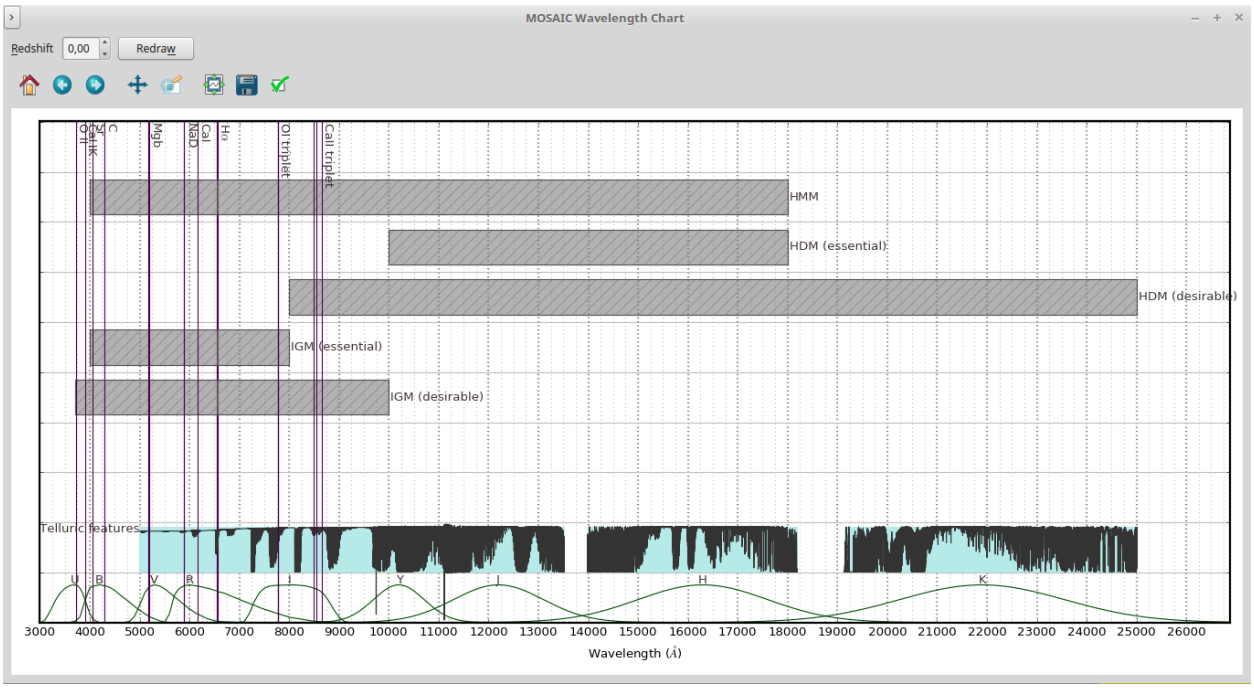


Figure – Lines with zero redshift

This application creates a chart stacking the MOSAIC spectrograph wavelength coverages and an ESO Earth atmospheric model. This may serve either as a reference to MOSAIC wavelength invervals for each mode (on this, see also `list-mosaic-modes.py`) or to verify the Earth atmospheric emission/trasmission in a wavelength region of observational interest.

It is also possible to inform a redshift so that the chemical lines will be accordingly displaced:

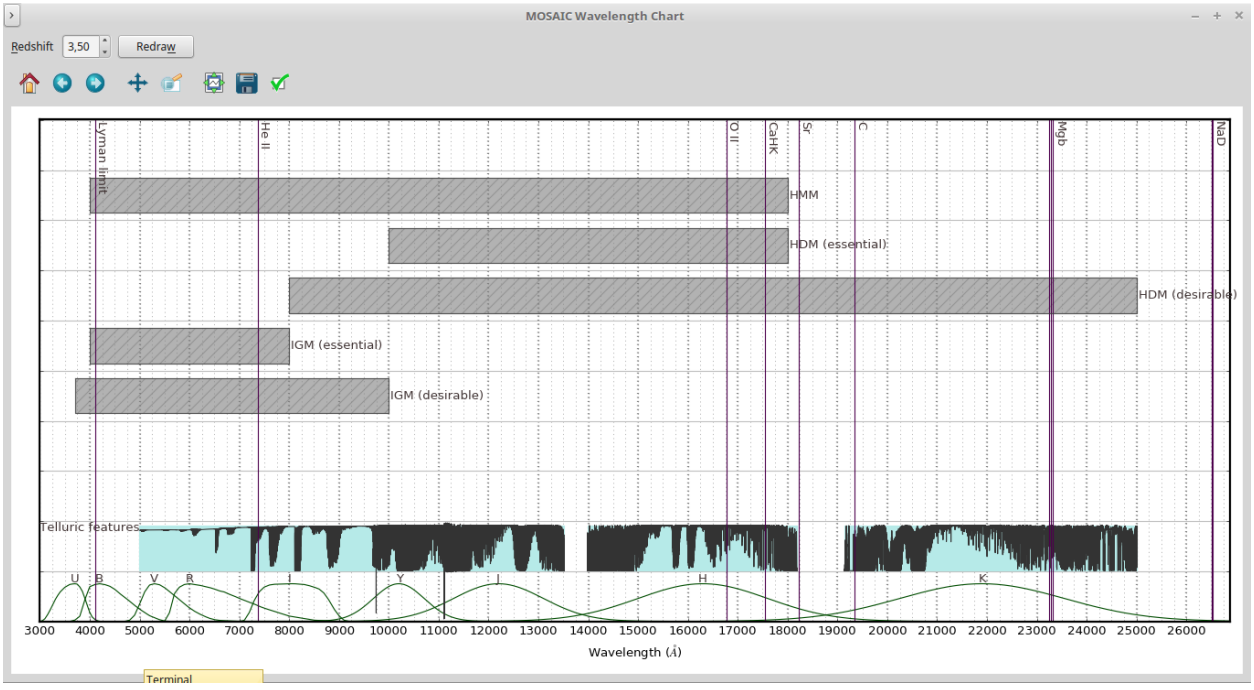


Figure – `z=3.5`

9.1.3 Download simulation results

The following example assumes that simulations coded from 1700 to 1721 already finished on the WebSim-COMPASS server.

`get-compass.py` is a Python script based on `get-compass.sh` which can be downloaded from the WebSim-COMPASS webpage. The former enhances the latter in which:

- It can download several simulations in a single command
- It is possible to specify the âĂIJstageâĂİ of the simulation pipeline to download results from. For example, it is possible to download only the âĂIJspintgâĂİ file, skipping the large data cubes from intermediary stages.

```
get-compass.py 1700-1721 --stage spintg
```

will download results for simulations *C001700*, *C001701*, âĂę, *C001721* **into the local directory**, after which you will see files `C*.fits`, `C*.par`, `C*.out`

9.1.4 Organize simulation results

Group resulting spectra in a single file

This step is required for later analysis using `splisted.py`

The following command will group all files âĂIJC*_spintg.fitsâĂİ into a single âĂIJ.splistâĂİ (Spectrum List) file, which can later be opened using `splisted.py`

```
$ create-spectrum-lists.py
.
.
.
[INFO    ] Created file './group-spintg-00-C001700-C001721.splist'
[INFO    ] Created file './group-spintg-01-C001712-C001712.splist'
```

Create reports (optional)

This step creates HTML pages (one for each simulation) that help to navigate through the simulation results.

```
create-simulation-reports.py 1700-1721
```

Organize the directory

At this point, the current directory has a large number of files (âĂIJ.fitsâĂİ, âĂIJ.htmlâĂİ, âĂIJ.pngâĂİ, etc.), whereas for our analysis, only the âĂIJ.splistâĂİ file is required.

`organize-directory.py` will:

- create a directory named âĂIJrawâĂİ where it will copy âĂIJ.fitsâĂİ, âĂIJ.parâĂİ and âĂIJ.outâĂİ files
- create a directory named âĂIJreportsâĂİ where it will copy âĂIJ.htmlâĂİ and âĂIJ.pngâĂİ files. In addition, it will create a file âĂIJindex.htmlâĂİ that will serve as an index for the âĂIJ.htmlâĂİ files

```
organize-directory.py
.
.
```

```
.
[INFO    ]   - Move 108 objects
[INFO    ]   - Create 'reports/index.html'
Continue (Y/n)?
```

9.1.5 Browse through reports

```
cd reports
xdg-open index.html
```

will open file âĂIJindex.htmlâĂİ in browser

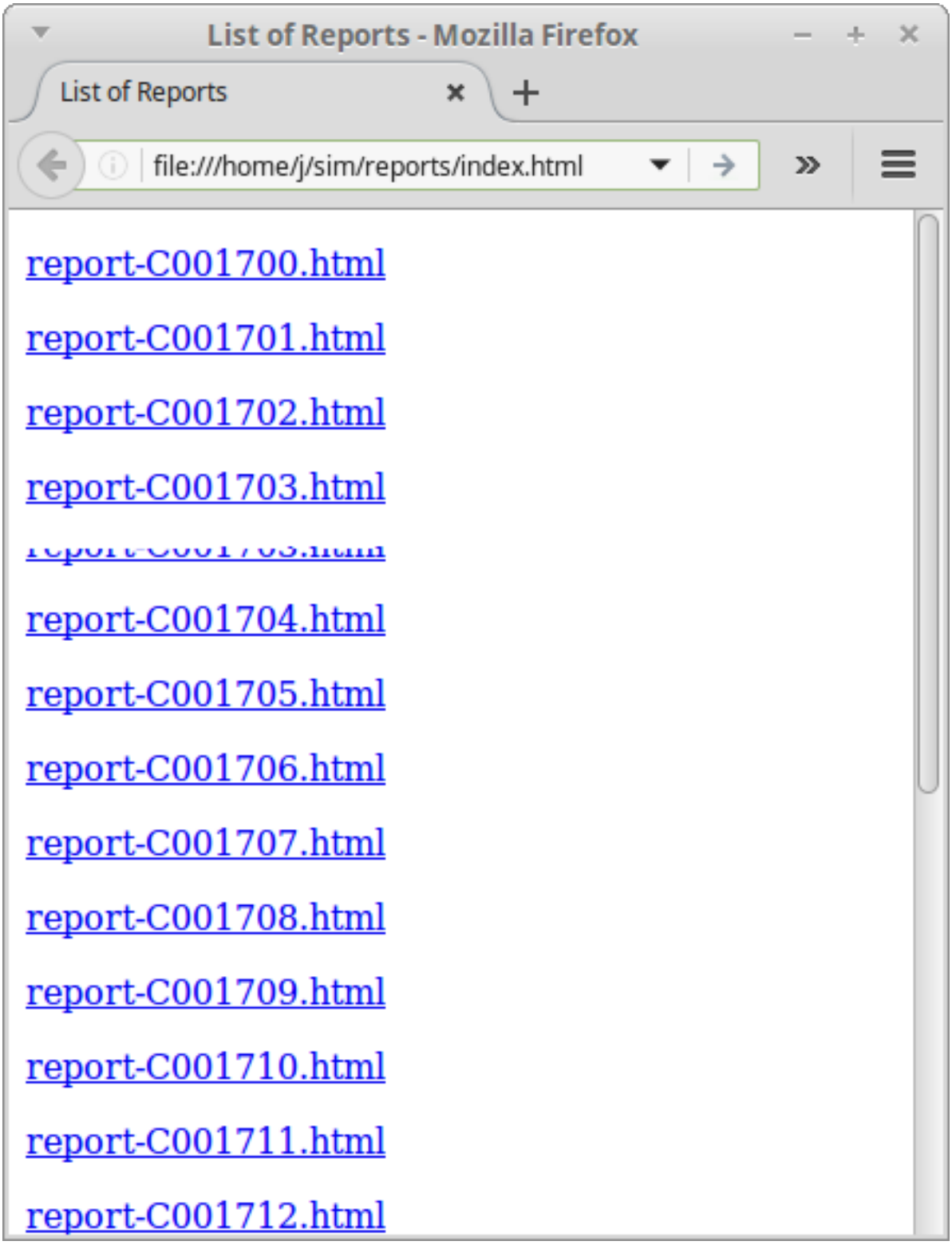


Figure – Reports index

9.1.6 Edit Spectrum List file

If you types the commands above to visualize reports, you will need to go back one directory level:

```
cd ..
```

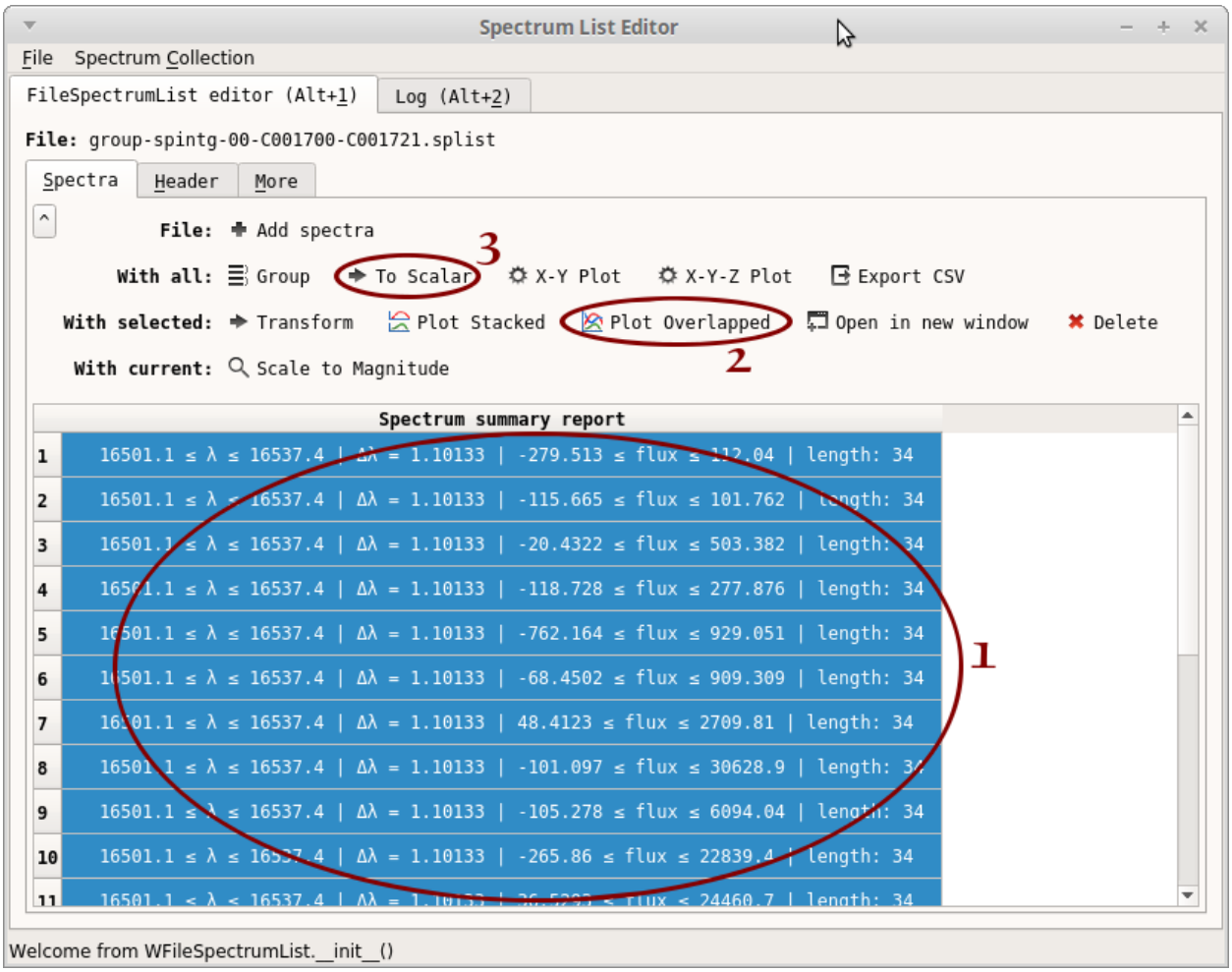
Now open the Spectrum List Editor (part of the f311 package):

```
splisted.py group-spintg-00-C001700-C001721.splist
```

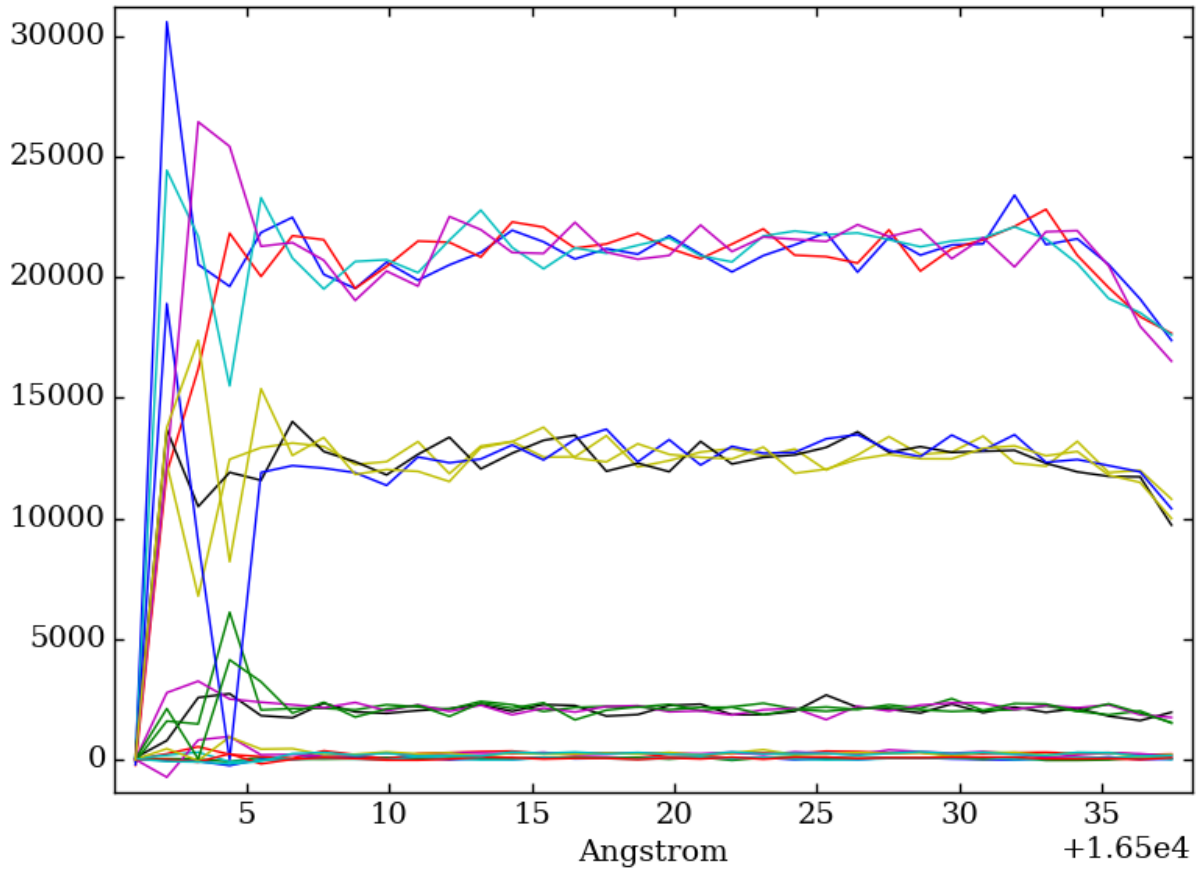
In the following steps, we will:

- Plot the spectra
- Calculate the Signal-to-noise ratio (SNR)
- Plot the Detector Integration Time (DIT) *vs* the SNR

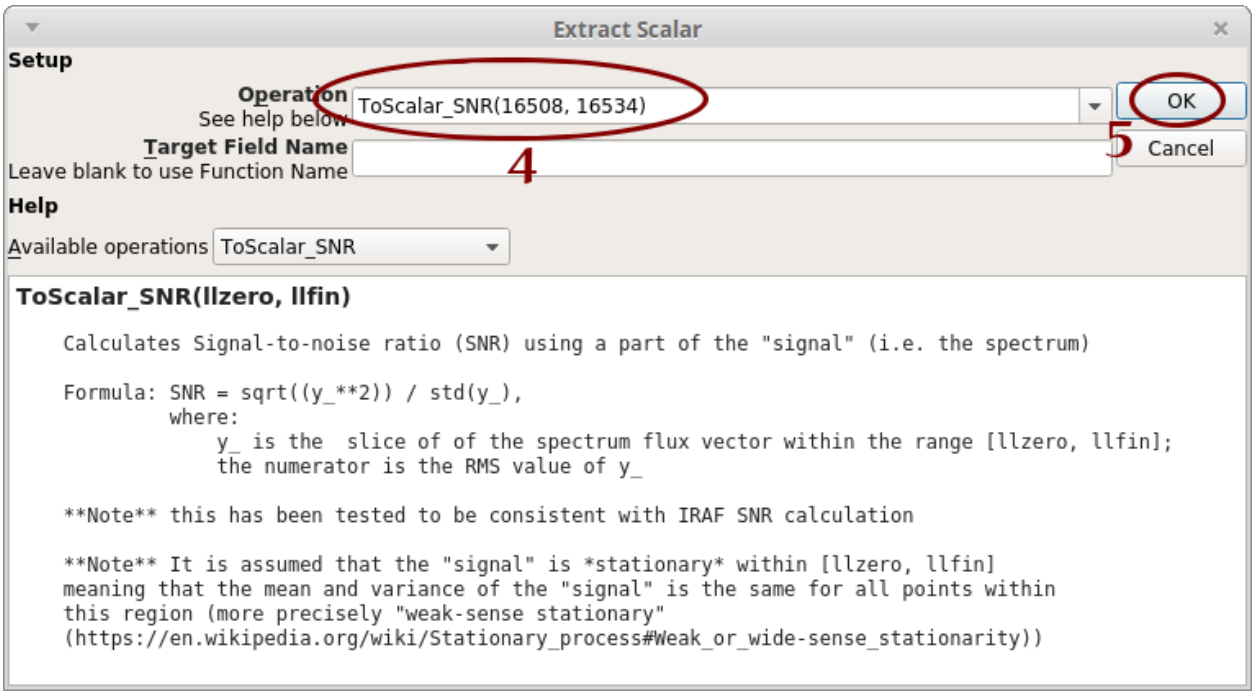
1. Select all the spectra: click inside the table, then press **Ctrl+A**



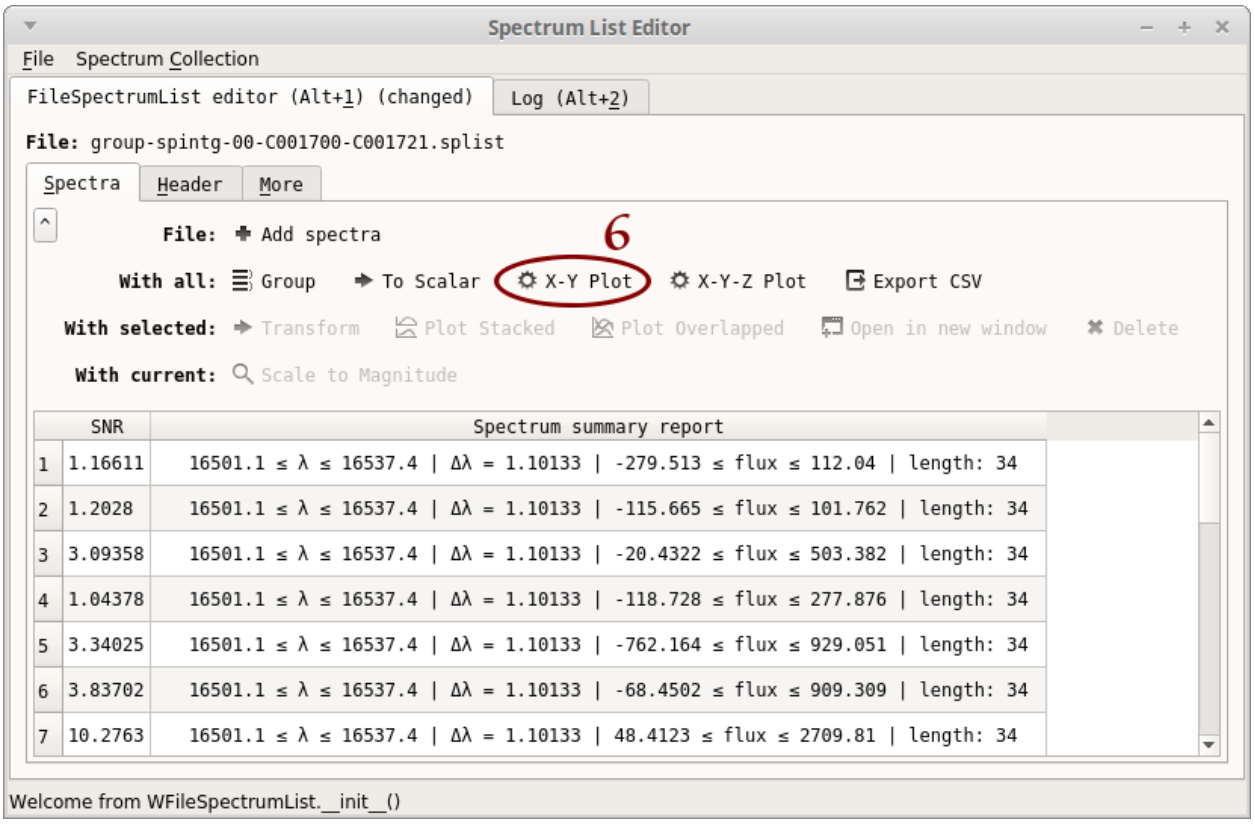
2. Click on âĂIJPlot OverlappedâĂİ. A plot window opens. From this plot, we can see that the region 16508-16534 seems to be free of atmospheric contamination. You may close the plot window



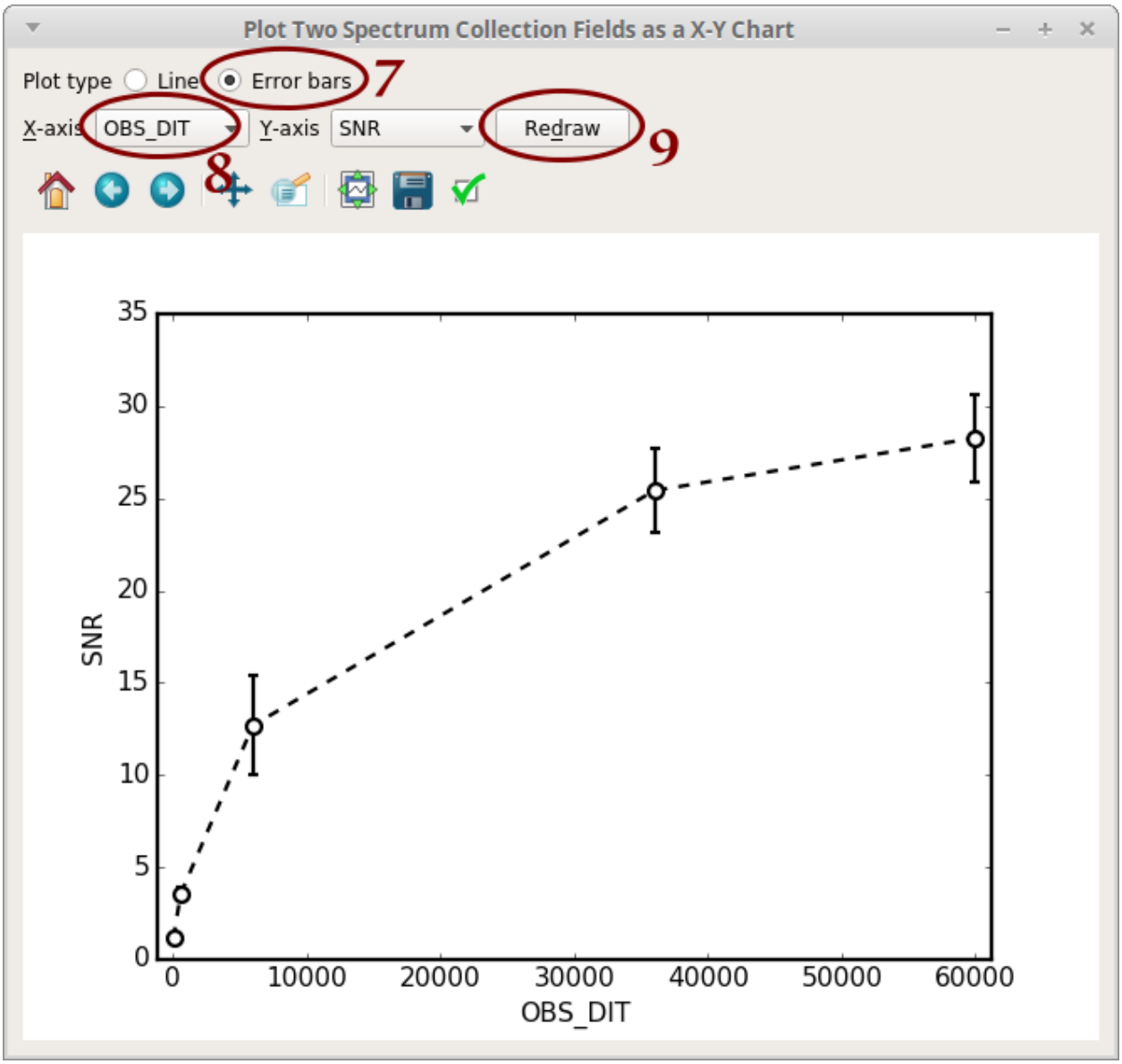
3. Click on âĂIJTo ScalarâĂİ. Another window opens
4. Type âĂIJToScalar_SNR(16508, 16534)âĂİ
5. Click on âĂIJOKâĂİ



6. Notice that a new column âĂIJSNRâĂİ appear in the table. Click on âĂIJX-Y PlotâĂİ



7. Select âĂIJError barsâĂİ
8. Select âĂIJOBS_DITâĂİ
9. Click on âĂIJRedrawâĂİ



9.2 API reference

autodoc/f311.aosss

CHAPTER

TEN

INDEX OF APPLICATIONS (SCRIPTS)

10.1 Script `create-simulation-reports.py`

```
usage: create-simulation-reports.py [-h] [--dir [DIR]] [--max N] N [N ...]

Creates HTML reports from WebSim-COMPASS output files

positional arguments:
  N             List of simulation numbers (single value and ranges accepted,
                e.g. 1004, 1004-1040)

optional arguments:
  -h, --help    show this help message and exit
  --dir [DIR]   Input directory (default: .)
  --max N       Maximum allowed number of reports (default: 100)
```

This script belongs to package *f311.aosss*

10.2 Script `create-spectrum-lists.py`

```
usage: create-spectrum-lists.py [-h] [--stage [STAGE]]

Create several .splist (spectrum list) files from WebSim-COMPASS output files; groups␣
↪spectra that share same wavelength vector

All spectra in each .splist file will have the same wavelength vector

optional arguments:
  -h, --help       show this help message and exit
  --stage [STAGE]  Websim-Compass pipeline stage (will collect files named,
                   e.g., C000793_<stage>.fits) (default: spintg)
```

This script belongs to package *f311.aosss*

10.3 Script `get-compass.py`

```
usage: get-compass.py [-h] [--max N] [--stage [STAGE]] N [N ...]

Downloads WebSim-COMPASS simulations
```

```
Based on shell script by Mathieu Puech

**Note** Skips simulations for existing files in local directory starting with
         that simulation ID.
         Example: if it finds file(s) "C001006*", will skip simulation C001006

**Note** Does not create any directory (actually creates it but deletes later).
         All files stored in local directory!

**Note** Will work only on if os.name == "posix" (Linux, UNIX ...)

positional arguments:
  N                 List of simulation numbers (single value and ranges
                    accepted, e.g. 1004, 1004-1040)

optional arguments:
  -h, --help        show this help message and exit
  --max N           Maximum number of simulations to get (default: 100)
  --stage [STAGE]   Websim-Compass pipeline stage: if specified, will download
                    files named, e.g., C000793_<stage>.fits (**note**: .par and
                    .out files are always downloaded) (default: all)
```

This script belongs to package *f311.aosss*

10.4 Script `list-mosaic-modes.py`

```
usage: list-mosaic-modes.py [-h] [search]

Lists MOSAIC Spectrograph modes

positional arguments:
  search      Search string (optional) (e.g., "HMM") (default: None)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.aosss*

10.5 Script `organize-directory.py`

```
usage: organize-directory.py [-h]

Organizes simulation directory (creates folders, moves files, creates 'index.html')

  - moves 'root/report-*'       to 'root/reports'
  - moves 'root/C*'             to 'root/raw'
  - moves 'root/raw/simgroup*'  to 'root/'
  - moves 'root/raw/report-*'   to 'root/reports'
  - moves 'root/raw/group*.splist'   to 'root'
  - [re]creates 'root/reports/index.html'

This script can be run from one of these directories:
```

```
  - 'root' -- a directory containing at least one of these directories: 'reports',
↪'raw'
  - 'root/raw'
  - 'root/reports'

The script will use some rules to try to figure out where it is running from

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.aosss*

10.6 Script `wavelength-chart.py`

```
usage: wavelength-chart.py [-h] [--plot]

Draws chart showing spectral lines of interest, spectrograph wavelength ranges, ESO␣
↪atmospheric model, etc.

Two modes are available:
  - GUI mode (default): opens a GUI allowing for setup parameters
  - Plot mode (--plot): plots the chart directly in default way

optional arguments:
  -h, --help  show this help message and exit
  --plot      Plot mode (default is GUI mode) (default: False)
```

This script belongs to package *f311.aosss*

10.7 Script `hitran-scraper.py`

```
usage: hitran-scraper.py [-h] [-t T] [M] [I] [llzero] [llfin]

Retrieves molecular lines from the HITRAN database [Gordon2016]

This script uses web scraping and the HAPI to save locally molecular lines from the␣
↪HITRAN database.

While the HAPI provides the downloading facility, web scraping is used to get the␣
↪lists of molecules
and isotopologues from the HITRAN webpages and get the IDs required to run the HAPI␣
↪query.

The script is typically invoked several times, each time with an additional argument.

References:

[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. CsÃąszÃąr, 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. MÃijller, 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. WcisÅĆo, S. Yu, E.J.␣
↪Zak,
    The HITRAN2016 Molecular Spectroscopic Database, J. Quant. Spectrosc. Radiat.␣
↪Transf. (2017).
    doi:10.1016/j.jqsrt.2017.06.038.

positional arguments:
  M           HITRAN molecule number (default: (lists molecules))
  I           HITRAN isotopologue number (not unique, starts over at each
              molecule) (default: (lists isotopologues))
  llzero      Initial wavelength (Angstrom) (default: None)
  llfin       Final wavelength (Angstrom) (default: None)

optional arguments:
  -h, --help  show this help message and exit
  -t T        Table Name (default: (molecular formula))
```

This script belongs to package *f311.convmol*

10.7.1 Usage examples

```
$ hitran-scraper.py

List of all HITRAN molecules
============================

  ID  Formula    Name
----  ---------  --------------------
   1  H2O        Water
   2  CO2        Carbon Dioxide
   3  O3         Ozone
   4  N2O        Nitrous Oxide
   5  CO         Carbon Monoxide
   6  CH4        Methane
   7  O2         Molecular Oxygen
   8  NO         Nitric Oxide
   9  SO2        Sulfur Dioxide
  10  NO2        Nitrogen Dioxide
  11  NH3        Ammonia
  12  HNO3       Nitric Acid
  13  OH         Hydroxyl Radical
  14  HF         Hydrogen Fluoride
  15  HCl        Hydrogen Chloride
  16  HBr        Hydrogen Bromide
  17  HI         Hydrogen Iodide
  18  ClO        Chlorine Monoxide
  19  OCS        Carbonyl Sulfide
  20  H2CO       Formaldehyde
```

```
  21  HOCl        Hypochlorous Acid
  22  N2          Molecular Nitrogen
  23  HCN         Hydrogen Cyanide
  24  CH3Cl       Methyl Chloride
  25  H2O2        Hydrogen Peroxide
  26  C2H2        Acetylene
  27  C2H6        Ethane
  28  PH3         Phosphine
  29  COF2        Carbonyl Fluoride
  31  H2S         Hydrogen Sulfide
  32  HCOOH       Formic Acid
  33  HO2         Hydroperoxyl Radical
  34  O           Oxygen Atom
  36  NO+         Nitric Oxide Cation
  37  HOBr        Hypobromous Acid
  38  C2H4        Ethylene
  39  CH3OH       Methanol
  40  CH3Br       Methyl Bromide
  41  CH3CN       Methyl Cyanide
  43  C4H2        Diacetylene
  44  HC3N        Cyanoacetylene
  45  H2          Molecular Hydrogen
  46  CS          Carbon Monosulfide
  47  SO3         Sulfur trioxide

Now, to list isotopologues for a given molecule, please type:

    hitran-scraper.py <molecule ID>

where <molecule ID> is one of the IDs listed above.
```

Now suppose we want the molecule OH molecule:

```
$ hitran-scraper.py 13

List of all isotopologues for molecule 'OH' (Hydroxyl Radical)
==============================================================

m_formula      ID    ID_molecule  Formula      AFGL_Code  Abundance
-----------  ----  -------------  ---------  -----------  ---------------
OH              1             13  (16)OH              61  0.997473
OH              2             13  (18)OH              81  0.002
OH              3             13  (16)OD              62  1.553710ÂăÃŮÂă10-4


Now, to download lines, please type:

    hitran-scraper.py 13 <isotopologue ID> <llzero> <llfin>

where <isotopologue ID> is one the numbers from the 'ID' column above,

and [<llzero>, <llfin>] defines the wavelength interval in Angstrom.
```

Now selecting the first isotopologue and specifying the visible wavelength range:

```
$ hitran-scraper.py 13 1 3000 7000

Isotopologue selected:
```

```
======================

Field name    Value
------------  --------
m_formula     OH
ID            1
ID_molecule   13
Formula       (16)OH
AFGL_Code     61
Abundance     0.997473

Wavelength interval (air): [3000.0, 7000.0] Angstrom
Wavenumber interval (vacuum): [14289.61969369552, 33342.42546386186] cm**-1
Table name: '(16)OH'

Fetching data...
===
=== BEGIN messages from HITRAN API ===
===
BEGIN DOWNLOAD: (16)OH
  65536 bytes written to ./(16)OH.data
  65536 bytes written to ./(16)OH.data
  65536 bytes written to ./(16)OH.data
  65536 bytes written to ./(16)OH.data
  65536 bytes written to ./(16)OH.data
  65536 bytes written to ./(16)OH.data
  65536 bytes written to ./(16)OH.data
  65536 bytes written to ./(16)OH.data
  65536 bytes written to ./(16)OH.data
  65536 bytes written to ./(16)OH.data
Header written to ./(16)OH.header
END DOWNLOAD
                     Lines parsed: 3855
PROCESSED
===
=== END messages from HITRAN API ===
===
...done
Please check files '(16)OH.header', '(16)OH.data'
```

10.7.2 Quick note on the HITRAN API

The files created (âĂŸ(16)OH.headerâĂŹ, âĂŸ(16)OH.dataâĂŹ) can be opened using the HAPI. They are also accessed by the application `convmol.py`.

The HAPI can be downloaded, but one version is also included with the f311 package. The following is an example of how the HITRAN data could be accessed from the Python console:

```
>>> from f311 import hapi
>>> hapi.loadCache()
Using .
(16)OH
                     Lines parsed: 3855
>>> oh_data = hapi.LOCAL_TABLE_CACHE["(16)OH"]
>>> oh_data.keys()
dict_keys(['data', 'header'])
```

```
>>> oh_data["data"].keys()
dict_keys(['ierr', 'gpp', 'molec_id', 'global_lower_quanta', 'sw', 'gamma_self', 'n_
↪air', 'elower', 'line_mixing_flag', 'local_lower_quanta', 'gp', 'global_upper_quanta
↪', 'gamma_air', 'local_upper_quanta', 'iref', 'local_iso_id', 'delta_air', 'nu', 'a
↪'])
```

To work properly with these data in your code, you may have a look at the HAPI source code and manual, as this library is superbly documented.

Within f311, the code in `f311.convmol.conv_hitran.hitran_to_sols()` contains a usage example of HITRAN data.

10.8 Script `nist-scraper.py`

```
usage: nist-scraper.py [-h] formula

Retrieves and prints a table of molecular constants from the NIST Chemistry Web Book␣
↪[NISTRef]

To do so, it uses web scraping to navigate through several pages and parse the␣
↪desired information
from the book web pages.

It does not provide a way to list the molecules yet, but will give an error if the␣
↪molecule is not
found in the NIST web book.

Example:

    print-nist.py OH

**Note** This script was designed to work with **diatomic molecules** and may not␣
↪work with other
         molecules.

**Warning** The source material online was known to contain mistakes (such as an␣
↪underscore instead
            of a minus signal to indicate a negative number). We have identified a␣
↪few of these,
            and build some workarounds. However, we recommend a close look at the␣
↪information parsed
            before use.

**Disclaimer** This script may stop working if the NIST people update the Chemistry␣
↪Web Book.

References:

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

positional arguments:
  formula     NIST formula

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.convmol*

10.8.1 Usage examples

Usage examples:

```
nist-scraper.py TiO
```

will print

```
*** titanium oxide ***

State       T_e            omega_e    omega_ex_e    omega_ey_e      B_e    alpha_e
↪gamma_e       D_e    beta_e      r_e  Trans.       nu_00  A
----------  -----------  ---------  ------------  ------------  -------  ---------  --
↪-------  --------  --------  -------  --------  --------  ---
D           31920.0        1040
↪                                      D âĘŤ X     31940
e ÂĄSigmaâĄž   a + 26598.1     845.2          4.2                  0.4892     0.0023
↪             4.7e-07             1.695    e âĘŤ d R   24297.5
f ÂĄDelta    (a + 19132)     890                                  0.50221
↪            6.4e-07             1.67292  f âĘŤ a R   19068.9
c ÂĄPhi      a + 17890.2     909.6          4.19                 0.523      0.00313
↪            3.9e-07             1.6393   c âĘŤ a R   17840.6
C 3Delta_r  19617.0         838.26         4.76         0.047   0.48989    0.00306   -
↪3e-05    6.7e-07             1.69383  C âĘŤ X R   19334
B 3Pi_r     16331.3         875            5                    0.50617
↪         6.86e-07            1.66636  B âĘŤ X R   16066.7
b ÂĄPi       a+11322.0_3     911.2          3.72                   0.51337    0.00291
↪            6.1e-07             1.65464  b âĘŤ d R    9054.02
A 3Phi_r    14431.0         867.78         3.942                0.50739    0.00315   -
↪1e-05    6.92e-07     2e-09  1.66436  A âĘŤ X R   14163
E 3Pi       12025.0         924.2          5.1
↪                                     E âĘŤ X     11871
d ÂĄSigmaâĄž   a + 2215.6     1014.6           4.64                 0.54922    0.
↪00337             6e-07               1.59972
a ÂĄDelta    a              1009.3          3.93                 0.5376     0.00298
↪           5.9e-07             1.61692
X 3Delta_r  197.5          1009.02         4.498       -0.0107  0.53541    0.00301   -
↪1.1e-05  6.03e-07     3e-09  1.62022
```

```
nist-scraper.py OH
```

will print

```
*** Hydroxyl radical ***

State          T_e    omega_e    omega_ex_e  omega_ey_e        B_e    alpha_e
↪gamma_e       D_e  beta_e        r_e  Trans.        nu_00        A
---------  -------  ---------  ------------  ------------  -------  ---------  -------
↪--  --------  --------  -------  ---------  --------  -------
C 2SigmaâĄž  89459.1    1232.9        19.1                   4.247      0.078
↪     0.0002             2.0461   C → A R    55820.7
C 2SigmaâĄž
↪                                  (C → X)    88223
D 2SigmaâĄż  82130      2954                                 15.2179
↪     0.001616           1.08093  D âĘŘ X R    81759.8
```

```
B ²SigmaâĄž  69774       660                                  5.086
↪     0.000929           1.8698   B → A R    35965.5
A ²SigmaâĄž  32684.1    3178.86       92.917                 17.358      0.7868     -0.
↪016  0.002039           1.0121   A âĘŤ X R    32402.4
X ²Pi_i        0      3737.76       84.8813               18.9108     0.7242
↪    0.001938            0.96966  1/2 âĘŘ 3/2    126.23  -139.21
```

10.9 Script `convmol.py`

```
usage: convmol.py [-h] [--fn_moldb [FN_MOLDB]] [--fn_molconsts [FN_MOLCONSTS]]
                  [--fn_config [FN_CONFIG]]

Conversion of molecular lines data to PFANT format

optional arguments:
  -h, --help            show this help message and exit
  --fn_moldb [FN_MOLDB]
                        File name for Database of Molecular Constants
                        (default: moldb.sqlite)
  --fn_molconsts [FN_MOLCONSTS]
                        File name for Molecular constants config file (Python
                        code) (default: configmolconsts.py)
  --fn_config [FN_CONFIG]
                        File name for Configuration file for molecular lines
                        conversion GUI (Python code) (default:
                        configconvmol.py)
```

This script belongs to package *f311.convmol*

10.10 Script `mced.py`

```
usage: mced.py [-h] [fn]

Editor for molecular constants file

This application can edit files of class FileMolConsts.

positional arguments:
  fn          Molecular constants file name (default: configmolconsts.py)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.convmol*

10.11 Script `moldbed.py`

```
usage: moldbed.py [-h] [fn]

Editor for molecules SQLite database
```

```
This application can edit files of class FileMolDB.

positional arguments:
  fn          Molecules database file name (default: moldb.sqlite)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.convmol*

10.12 Script `create-grid.py`

```
usage: create-grid.py [-h] [--pattern [PATTERN]]
                      [--mode [{opa,modtxt,modbin}]]
                      [fn_output]

Merges several atmospheric models into a single file (_i.e._, the "grid")

"Collects" several files in current directory and creates a single file
containing atmospheric model grid.

Working modes (option "-m"):
 "opa" (default mode): looks for MARCS[1] ".mod" and ".opa" text file pairs and
                       creates a *big* binary file containing *all* model
                       information including opacities.
                       Output will be in ".moo" format.

 "modtxt": looks for MARCS ".mod" text files only. Resulting grid will not contain
           opacity information.
           Output will be in binary ".mod" format.

 "modbin": looks for binary-format ".mod" files. Resulting grid will not contain
           opacity information.
           Output will be in binary ".mod" format.

References:
  [1] http://marcs.astro.uu.se/

.
.
.

positional arguments:
  fn_output             output file name (default: "grid.moo" or "grid.mod",
                        depending on mode)

optional arguments:
  -h, --help            show this help message and exit
  --pattern [PATTERN]   file name pattern (with wildcards) (default: *.mod)
  --mode [{opa,modtxt,modbin}]
                        working mode (see description above) (default: opa)
```

This script belongs to package *f311.explorer*

10.13 Script `cut-atoms.py`

```
usage: cut-atoms.py [-h] llzero llfin fn_input fn_output

Cuts atomic lines file to wavelength interval specified

The interval is [llzero, llfin]

positional arguments:
  llzero      lower wavelength boundary (angstrom)
  llfin       upper wavelength boundary (angstrom)
  fn_input    input file name
  fn_output   output file name

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.14 Script `cut-molecules.py`

```
usage: cut-molecules.py [-h] llzero llfin fn_input fn_output

Cuts molecular lines file to wavelength interval specified

The interval is [llzero, llfin]

positional arguments:
  llzero      lower wavelength boundary (angstrom)
  llfin       upper wavelength boundary (angstrom)
  fn_input    input file name
  fn_output   output file name

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.15 Script `cut-spectrum.py`

```
usage: cut-spectrum.py [-h] llzero llfin fn_input fn_output

Cuts spectrum file to wavelength interval specified

Resulting spectrum Saved in 2-column ASCII format

The interval is [llzero, llfin]

positional arguments:
  llzero      lower wavelength boundary (angstrom)
  llfin       upper wavelength boundary (angstrom)
  fn_input    input file name
  fn_output   output file name
```

```
optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.16 Script `plot-spectra.py`

```
usage: plot-spectra.py [-h] [--ovl | --pieces | --pages] [--aint [AINT]]
                       [--fn_output [FN_OUTPUT]] [--ymin [YMIN]]
                       [-r [NUM_ROWS]]
                       fn [fn ...]

Plots spectra on screen or creates PDF file

It can work in four different modes:

a) grid of sub-plots, one for each spectrum (default mode)
   Example:
   plot-spectra.py flux.norm.nulbad measured.fits

b) single plot with all spectra overlapped ("--ovl" option)
   Example:
   > plot-spectra.py --ovl flux.norm.nulbad measured.fits

c) PDF file with a small wavelength interval per page ("--pieces" option).
   This is useful to flick through a large wavelength range.
   Example:
   > plot-spectra.py --pieces --aint 7 flux.norm.nulbad measured.fits

d) PDF file with one spectrum per page ("--pages" option).
   Example:
   > plot-spectra.py --pages flux.*

Types of files supported:

  - pfant output, e.g., flux.norm;
  - nulbad output, e.g., flux.norm.nulbad;
  - 2-column "lambda-flux" generic text files;
  - FITS files.

positional arguments:
  fn                    name of spectrum file(s) (many types supported)
                        (wildcards allowed, e.g., "flux.*")

optional arguments:
  -h, --help            show this help message and exit
  --ovl                 Overlapped graphics (default: False)
  --pieces              If set, will generate a PDF file with each page
                        containing one "piece" of the spectra of lengthgiven
                        by the --aint option. (default: False)
  --pages               If set, will generate a PDF file with one spectrum per
                        page (default: False)
  --aint [AINT]         length of each piece-plot in wavelength units (used
                        only if --pieces) (default: 10)
  --fn_output [FN_OUTPUT]
```

```
                        PDF output file name (used only if --pieces) (default:
                        (plot-spectra-<xxxx>.pdf))
  --ymin [YMIN]         Minimum value for y-axis (default: (automatic))
  -r [NUM_ROWS], --num_rows [NUM_ROWS]
                        Number of rows in subplot grid (default: (automatic))
```

This script belongs to package *f311.explorer*

10.17 Script `vald3-to-atoms.py`

```
usage: vald3-to-atoms.py [-h] [--min_algf [MIN_ALGF]] [--max_kiex [MAX_KIEX]]
                         fn_input [fn_output]

Converts VALD3 atomic/molecular lines file to PFANT atomic lines file.

Molecular lines are skipped.

positional arguments:
  fn_input              input file name
  fn_output             output file name (default: atoms-untuned-<fn_input>)

optional arguments:
  -h, --help            show this help message and exit
  --min_algf [MIN_ALGF]
                        minimum algf (log gf) (default: -7)
  --max_kiex [MAX_KIEX]
                        maximum kiex (default: 15)
```

This script belongs to package *f311.explorer*

10.18 Script `abed.py`

```
usage: abed.py [-h] [fn]

Abundances file editor

positional arguments:
  fn          abundances file name (default: abonds.dat)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.19 Script `ated.py`

```
usage: ated.py [-h] [fn]

Atomic lines file editor

positional arguments:
```

```
  fn          atoms file name (default: atoms.dat)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.20 Script `cubeed.py`

```
usage: cubeed.py [-h] [fn]

Data Cube Editor, import/export WebSim-COMPASS data cubes

positional arguments:
  fn          file name, supports 'FITS Sparse Data Cube (storage to take less
              disk space)' and '' (default: None)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.21 Script `explorer.py`

```
usage: explorer.py [-h] [dir]

F311 Explorer --  list, visualize, and edit data files (_Ãă la_ File Manager)

positional arguments:
  dir         directory name (default: .)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.22 Script `mained.py`

```
usage: mained.py [-h] [fn]

Main configuration file editor.

positional arguments:
  fn          main configuration file name (default: main.dat)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.23 Script `mled.py`

```
usage: mled.py [-h] [fn]

Molecular lines file editor.

positional arguments:
  fn          molecules file name (default: molecules.dat)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.24 Script `optionsed.py`

```
usage: optionsed.py [-h] [fn]

PFANT command-line options file editor.

positional arguments:
  fn          PFANT Command-line Options file name (default: options.py)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.25 Script `splisted.py`

```
usage: splisted.py [-h] [fn]

Spectrum List Editor

positional arguments:
  fn          file name, supports 'FITS Spectrum List' only at the moment
              (default: None)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.explorer*

10.26 Script `tune-zinf.py`

```
usage: tune-zinf.py [-h] [--min [MIN]] [--max [MAX]] [--inflate [INFLATE]]
                    [--ge_current] [--no_clean]
                    fn_input [fn_output]

Tunes the "zinf" parameter for each atomic line in atomic lines file
```

```
The "zinf" parameter is a distance in angstrom from the centre of an atomic
line. It specifies the calculation range for the line:
[centre-zinf, centre+zinf].

This script runs pfant for each atomic line to determine the width of each
atomic line and thus zinf.

Note: pfant is run using most of its default settings and will require the
following files to exist in the current directory:
  - main.dat
  - dissoc.dat
  - abonds.dat
  - modeles.mod
  - partit.dat
  - absoru2.dat

Note: the precision in the zinf found depends on the calculation step ("pas")
specified in main.dat. A higher "pas" means lower precision and a tendency to
get higher zinf's. This is really not critical. pas=0.02 or pas=0.04 should do.

positional arguments:
  fn_input              input file name
  fn_output             output file name (default: <made-up filename>)

optional arguments:
  -h, --help            show this help message and exit
  --min [MIN]           minimum zinf. If zinf found for a particular line is
                        smaller than this value, this value will be used
                        instead (default: 0.1)
  --max [MAX]           maximum zinf. If zinf found for a particular line is
                        greater than this value, this value will be used
                        instead (default: 50.0)
  --inflate [INFLATE]   Multiplicative constant to apply a "safety margin".
                        Each zinf found will be multiplied by this value. For
                        example a value of INFLATE=1.1 means that all the
                        zinf's saved will be 10 percent larger than those
                        calculated (default: 1.1)
  --ge_current          "Greater or Equal to current": If this option is set,
                        the current zinf in the atomic lines file is used as a
                        lower boundary. (default: False)
  --no_clean            If set, will not remove the session directories.
                        (default: False)
```

This script belongs to package *f311.explorer*

10.27 Script `copy-star.py`

```
usage: copy-star.py [-h] [-l] [-p] [directory]

Copies stellar data files (such as main.dat, abonds.dat, dissoc.dat) to local␣
↪directory

examples of usage:
  > copy-star.py
  (displays menu)
```

```
  > copy-star.py arcturus
  ("arcturus" is the name of a subdirectory of PFANT/data)

  > copy-star.py -p /home/user/pfant-common-data
  (use option "-p" to specify path)

  > copy-star.py -l
  (lists subdirectories of PFANT/data , doesn't copy anything)

positional arguments:
  directory   name of directory (either a subdirectory of PFANT/data or the
              path to a valid system directory (see modes of operation)
              (default: None)

optional arguments:
  -h, --help  show this help message and exit
  -l, --list  lists subdirectories of
              /home/j/Documents/projects/astro/github/PFANT/code/data
              (default: False)
  -p, --path  system path mode (default: False)
```

This script belongs to package *f311.pyfant*

10.28 Script `link.py`

```
usage: link.py [-h] [-l] [-p] [-y] [directory]

Creates symbolic links to PFANT data files as an alternative to copying these␣
↪(sometimes large) files into local directory

A star is specified by three data files whose typical names are:
main.dat, abonds.dat, and dissoc.dat .

The other data files (atomic/molecular lines, partition function, etc.)
are star-independent, and this script is a proposed solution to keep you from
copying these files for every new case.

How it works: link.py will look inside a given directory and create
symbolic links to files *.dat and *.mod.

The following files will be skipped:
  - main files, e.g. "main.dat"
  - dissoc files, e.g., "dissoc.dat"
  - abonds files, e.g., "abonds.dat"
  - .mod files with a single model inside, e.g., "modeles.mod"
  - hydrogen lines files, e.g., "thalpha", "thbeta"

This script works in two different modes:

a) default mode: looks for files in a subdirectory of PFANT/data
   > link.py common
   (will create links to filess inside PFANT/data/common)

b) "-l" option: lists subdirectories of PFANT/data
```

```
c) "-p" option: looks for files in a directory specified.
   Examples:
   > link.py -p /home/user/pfant-common-data
   > link.py -p ../../pfant-common-data

Note: in Windows, this script must be run as administrator.

positional arguments:
  directory   name of directory (either a subdirectory of PFANT/data or the
              path to a valid system directory (see modes of operation)
              (default: common)

optional arguments:
  -h, --help  show this help message and exit
  -l, --list  lists subdirectories of
              /home/j/Documents/projects/astro/github/PFANT/code/data
              (default: False)
  -p, --path  system path mode (default: False)
  -y, --yes   Automatically answers 'yes' to eventual question (default:
              False)
```

This script belongs to package *f311.pyfant*

10.29 Script `merge-molecules.py`

```
usage: merge-molecules.py [-h] [-o [FN_OUTPUT]] files [files ...]

Merges several PFANT molecular lines file into a single one

positional arguments:
  files                 files specification: list of files, wildcards allowed

optional arguments:
  -h, --help            show this help message and exit
  -o [FN_OUTPUT], --fn_output [FN_OUTPUT]
                        output filename. If not specified, creates file such
                        as 'molecules-merged-.0000.dat' (default: (automatic))
```

This script belongs to package *f311.pyfant*

10.30 Script `run-multi.py`

```
usage: run-multi.py [-h] [--abs ABS] [--absoru ABSORU] [--aint AINT]
                    [--allow ALLOW] [--amores AMORES] [--convol CONVOL]
                    [--explain EXPLAIN] [--flam FLAM] [--flprefix FLPREFIX]
                    [--fn_abonds FN_ABONDS] [--fn_absoru2 FN_ABSORU2]
                    [--fn_atoms FN_ATOMS] [--fn_cv FN_CV]
                    [--fn_dissoc FN_DISSOC] [--fn_flux FN_FLUX]
                    [--fn_hmap FN_HMAP] [--fn_lines FN_LINES]
                    [--fn_log FN_LOG] [--fn_logging FN_LOGGING]
                    [--fn_main FN_MAIN] [--fn_modeles FN_MODELES]
                    [--fn_modgrid FN_MODGRID] [--fn_molecules FN_MOLECULES]
                    [--fn_moo FN_MOO] [--fn_opa FN_OPA]
```

```
                    [--fn_partit FN_PARTIT] [--fn_progress FN_PROGRESS]
                    [--fwhm FWHM] [--interp INTERP] [--kik KIK] [--kq KQ]
                    [--llfin LLFIN] [--llzero LLZERO]
                    [--logging_console LOGGING_CONSOLE]
                    [--logging_file LOGGING_FILE]
                    [--logging_level LOGGING_LEVEL] [--no_atoms NO_ATOMS]
                    [--no_h NO_H] [--no_molecules NO_MOLECULES] [--norm NORM]
                    [--opa OPA] [--pas PAS] [--pat PAT] [--play PLAY]
                    [--sca SCA] [--zinf ZINF] [--zph ZPH] [-f FN_ABXFWHM]
                    [-s CUSTOM_SESSION_ID]

Runs pfant and nulbad in "multi mode" (equivalent to Tab 4 in ``x.py``) (several␣
↪abundances X FWHM's)

optional arguments:
  -h, --help            show this help message and exit
  --abs ABS
  --absoru ABSORU
  --aint AINT
  --allow ALLOW
  --amores AMORES
  --convol CONVOL
  --explain EXPLAIN
  --flam FLAM
  --flprefix FLPREFIX
  --fn_abonds FN_ABONDS
  --fn_absoru2 FN_ABSORU2
  --fn_atoms FN_ATOMS
  --fn_cv FN_CV
  --fn_dissoc FN_DISSOC
  --fn_flux FN_FLUX
  --fn_hmap FN_HMAP
  --fn_lines FN_LINES
  --fn_log FN_LOG
  --fn_logging FN_LOGGING
  --fn_main FN_MAIN
  --fn_modeles FN_MODELES
  --fn_modgrid FN_MODGRID
  --fn_molecules FN_MOLECULES
  --fn_moo FN_MOO
  --fn_opa FN_OPA
  --fn_partit FN_PARTIT
  --fn_progress FN_PROGRESS
  --fwhm FWHM
  --interp INTERP
  --kik KIK
  --kq KQ
  --llfin LLFIN
  --llzero LLZERO
  --logging_console LOGGING_CONSOLE
  --logging_file LOGGING_FILE
  --logging_level LOGGING_LEVEL
  --no_atoms NO_ATOMS
  --no_h NO_H
  --no_molecules NO_MOLECULES
  --norm NORM
  --opa OPA
  --pas PAS
```

```
  --pat PAT
  --play PLAY
  --sca SCA
  --zinf ZINF
  --zph ZPH
  -f FN_ABXFWHM, --fn_abxfwhm FN_ABXFWHM
                        Name of file specifying different abundances and
                        FWHM's (default: abxfwhm.py)
  -s CUSTOM_SESSION_ID, --custom_session_id CUSTOM_SESSION_ID
                        Name of directory where output files will be saved
                        (default: multi-session-<i>)
```

This script belongs to package *f311.pyfant*

10.31 Script `run4.py`

```
usage: run4.py [-h] [--abs ABS] [--absoru ABSORU] [--aint AINT]
               [--allow ALLOW] [--amores AMORES] [--convol CONVOL]
               [--explain EXPLAIN] [--flam FLAM] [--flprefix FLPREFIX]
               [--fn_abonds FN_ABONDS] [--fn_absoru2 FN_ABSORU2]
               [--fn_atoms FN_ATOMS] [--fn_cv FN_CV] [--fn_dissoc FN_DISSOC]
               [--fn_flux FN_FLUX] [--fn_hmap FN_HMAP] [--fn_lines FN_LINES]
               [--fn_log FN_LOG] [--fn_logging FN_LOGGING] [--fn_main FN_MAIN]
               [--fn_modeles FN_MODELES] [--fn_modgrid FN_MODGRID]
               [--fn_molecules FN_MOLECULES] [--fn_moo FN_MOO]
               [--fn_opa FN_OPA] [--fn_partit FN_PARTIT]
               [--fn_progress FN_PROGRESS] [--fwhm FWHM] [--interp INTERP]
               [--kik KIK] [--kq KQ] [--llfin LLFIN] [--llzero LLZERO]
               [--logging_console LOGGING_CONSOLE]
               [--logging_file LOGGING_FILE] [--logging_level LOGGING_LEVEL]
               [--no_atoms NO_ATOMS] [--no_h NO_H]
               [--no_molecules NO_MOLECULES] [--norm NORM] [--opa OPA]
               [--pas PAS] [--pat PAT] [--play PLAY] [--sca SCA] [--zinf ZINF]
               [--zph ZPH]

Runs the four Fortran binaries in sequence: `innewmarcs`, `hydro2`, `pfant`, `nulbad`

Check session directory "session-<number>" for log files.

optional arguments:
  -h, --help            show this help message and exit
  --abs ABS
  --absoru ABSORU
  --aint AINT
  --allow ALLOW
  --amores AMORES
  --convol CONVOL
  --explain EXPLAIN
  --flam FLAM
  --flprefix FLPREFIX
  --fn_abonds FN_ABONDS
  --fn_absoru2 FN_ABSORU2
  --fn_atoms FN_ATOMS
  --fn_cv FN_CV
  --fn_dissoc FN_DISSOC
  --fn_flux FN_FLUX
```

```
  --fn_hmap FN_HMAP
  --fn_lines FN_LINES
  --fn_log FN_LOG
  --fn_logging FN_LOGGING
  --fn_main FN_MAIN
  --fn_modeles FN_MODELES
  --fn_modgrid FN_MODGRID
  --fn_molecules FN_MOLECULES
  --fn_moo FN_MOO
  --fn_opa FN_OPA
  --fn_partit FN_PARTIT
  --fn_progress FN_PROGRESS
  --fwhm FWHM
  --interp INTERP
  --kik KIK
  --kq KQ
  --llfin LLFIN
  --llzero LLZERO
  --logging_console LOGGING_CONSOLE
  --logging_file LOGGING_FILE
  --logging_level LOGGING_LEVEL
  --no_atoms NO_ATOMS
  --no_h NO_H
  --no_molecules NO_MOLECULES
  --norm NORM
  --opa OPA
  --pas PAS
  --pat PAT
  --play PLAY
  --sca SCA
  --zinf ZINF
  --zph ZPH
```

This script belongs to package *f311.pyfant*

10.32 Script `x.py`

```
usage: x.py [-h]

PFANT Launcher -- Graphical Interface for Spectral Synthesis

Single and multi modes.

Multi mode
----------

Runs pfant for different abundances for each element, then run nulbad for each
pfant result for different FWHMs.

The configuration is read from a .py file.

The user must specify a list of FWHM values for nulbad convolutions, and
a dictionary containing element symbols and respective list containing n_abdif
differential abundances to be used for each element.

pfant will be run n_abdif times, each time adding to each element in ab the i-th
```

```
value in the vector for the corresponding element.

nulbad will run n_abdif*n_fwhms times, where n_fwhms is the number of different
FWHMs specified.

The result will be
- several spectra saved as  "<star name><pfant name or counter>.sp"
- several "spectra list" files saved as "cv_<FWHM>.spl". As the file indicates,
  each ".spl" file will have the names of the spectrum files for a specific FWHM.
  .spl files are subject to input for lineplot.py by E.Cantelli
---------

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311.pyfant*