AOSSS Documentation

Release 17.12.21

Julio Trevisan

Dec 21, 2017

CONTENTS

Welcome!

CHAPTER

ONE

INTRODUCTION

Project AOSSS was started to support telescope+spectrograph simulations carried out at the WebSim-COMPASS plaftorm.

It provides tools to:

- create input data cubes for that plataform
- download, visualize, generate reports, and organize results from the simulator

In addition, there is a tool to plot spectral lines of interest *versus* spectrograph modes coverage at given redshifts.

The package also contains a library (API - application programming interface) to deal with photometric problems and to calculate the color of a spectrum. This API is used in the project applications, but may be used more broadly.

1.1 Acknowledgement

Funded by FAPESP - Research Support Foundation of the State of São Paulo, Brazil (2016-2017).

1.2 Contact

For bugs reports, questions, suggestions, etc., please open an issue at the project site on GitHub: http://github.com/trevisanj/aosss.

CHAPTER

TWO

INSTALLATION

If you have **Python 3** installed, then simply type:

```
pip install aosss
```

2.1 Pre-requisites

2.1.1 Python 3

If you need to set up your Python 3 environment, one option is to visit project F311 installation instructions at http://trevisanj.github.io/f311/install.html. That page also provides a troubleshooting section that applies.

2.2 Installing AOSSS in developer mode

This is an alternative to the "pip" at the beginning of this section. Use this option if you would like to download and modify the Python source code.

First clone the "aosss" GitHub repository:

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

or

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

Then, install AOSSS in **developer** mode:

```
cd aosss
python setup.py develop
```

2.3 Upgrade aosss

Package aosss can be upgraded to a new version by typing:

```
pip install aosss --upgrade
```

CHAPTER

THREE

PROCESSING SIMULATION RESULTS

3.1 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 "stage" of the simulation pipeline to download results from. For example, it is possible to download only the "spintg" file, skipping the large data cubes from intermediary stages.

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

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

3.2 Organize simulation results

3.2.1 Group resulting spectra in a single file

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

The following command will group all files "C*_spintg.fits" into a single ".splist" (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'
```

3.2.2 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
```

3.2.3 Organize the directory

At this point, the current directory has a large number of files (".fits", ".html", ".png", etc.), whereas for our analysis, only the ".splist" file is required.

`organize-directory.py` will:

- create a directory named "raw" where it will copy ".fits", ".par" and ".out" files
- create a directory named "reports" where it will copy ".html" and ".png" files. In addition, it will create a file "index.html" that will serve as an index for the ".html" files

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

3.3 Browse through reports

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

will open file "index.html" in browser

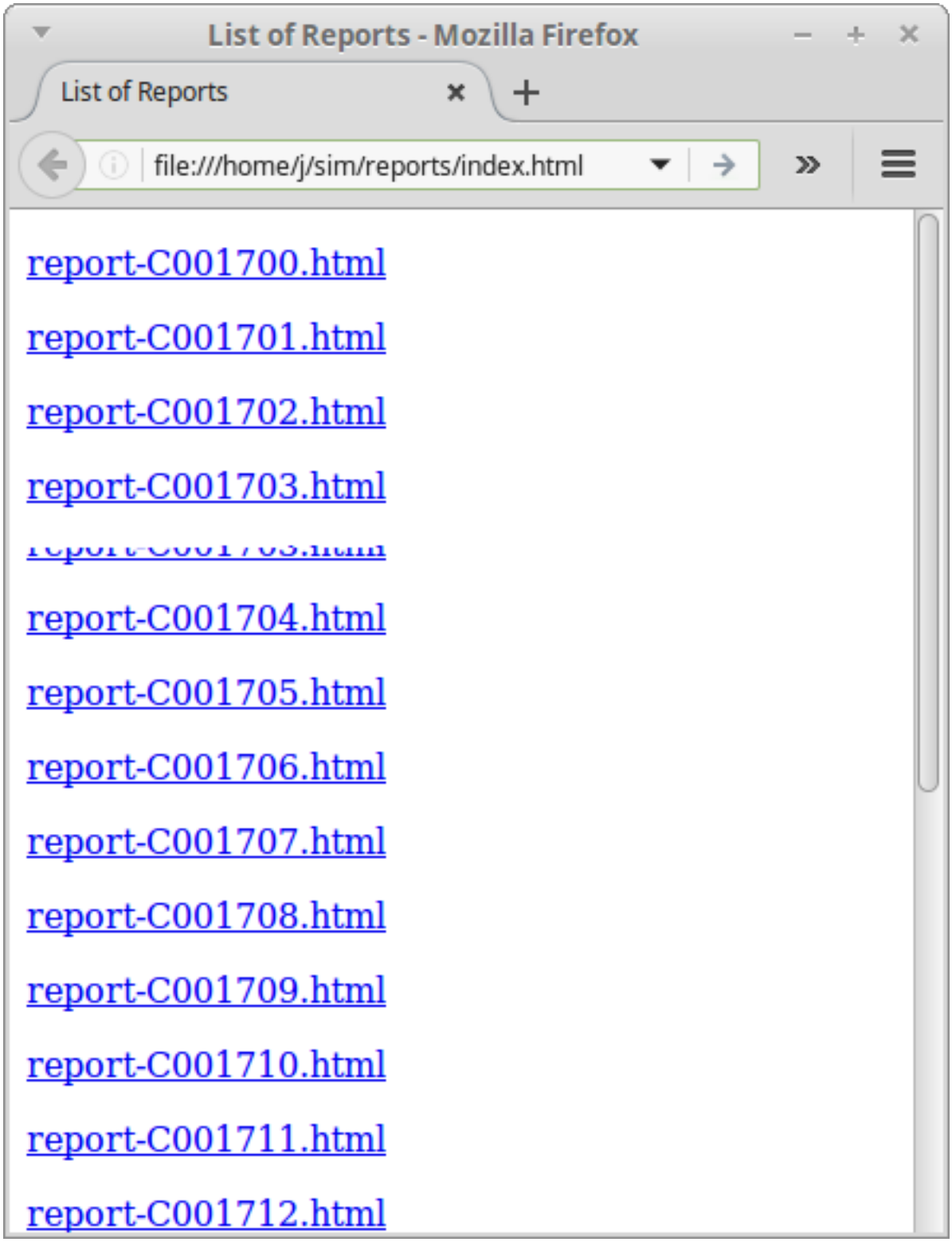


Figure – Reports index

3.4 Edit Spectrum List file

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

```
cd ..
```

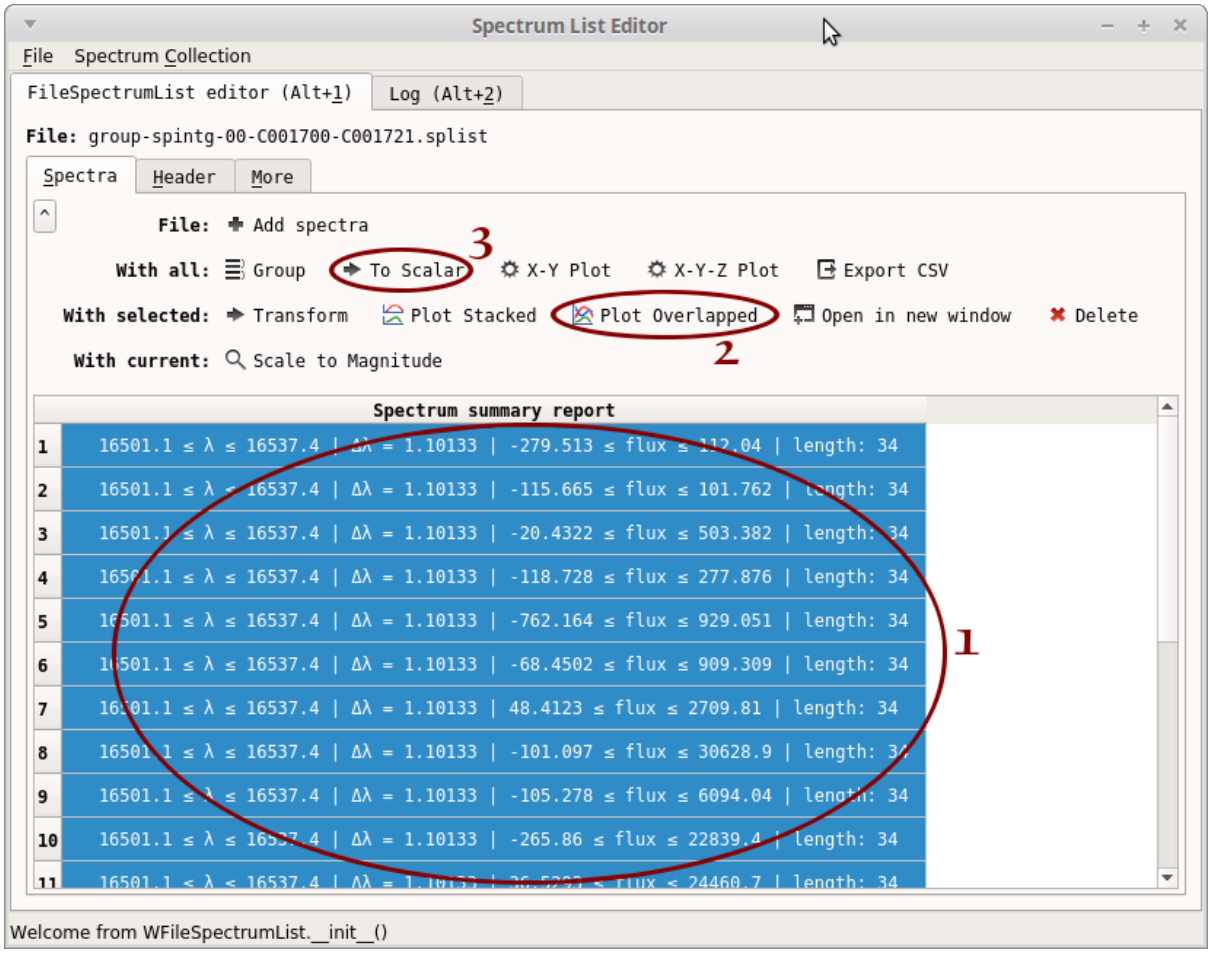
Now open the Spectrum List Editor (part of the f311 package):

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

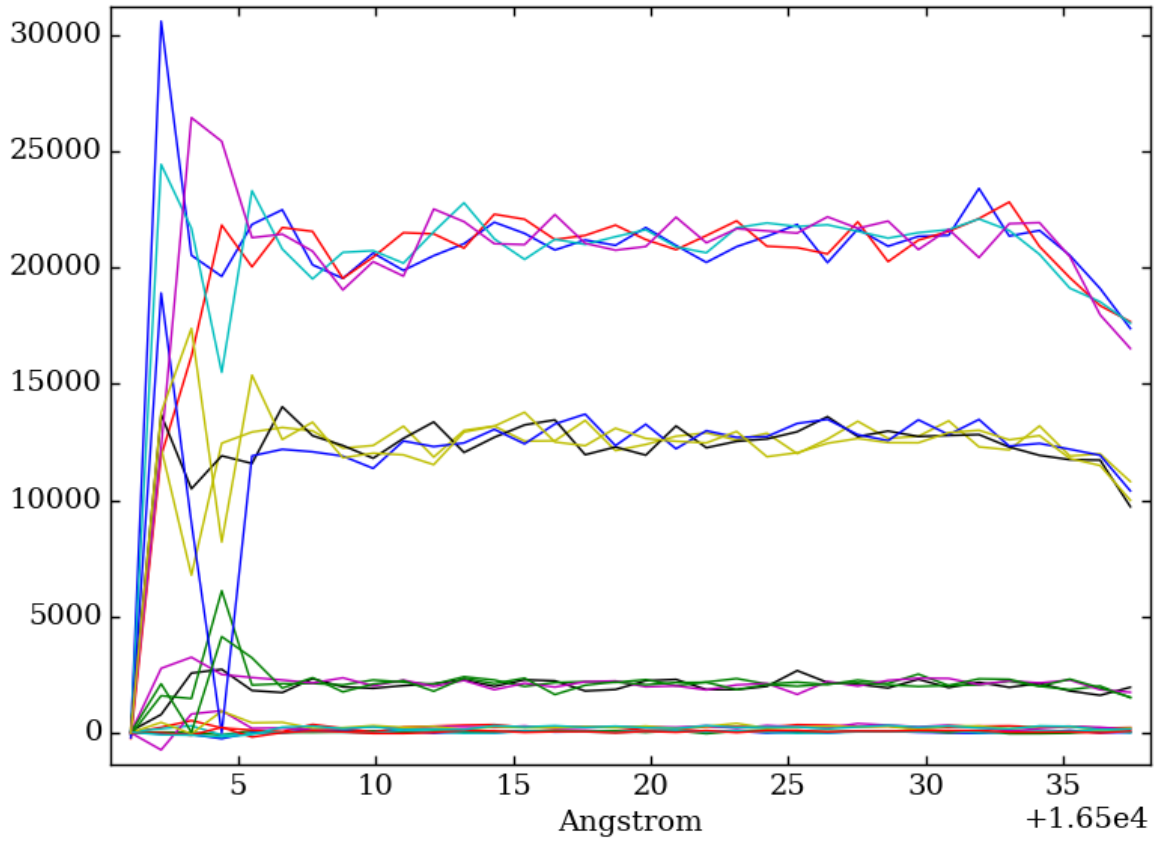
In the following steps, we will:

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

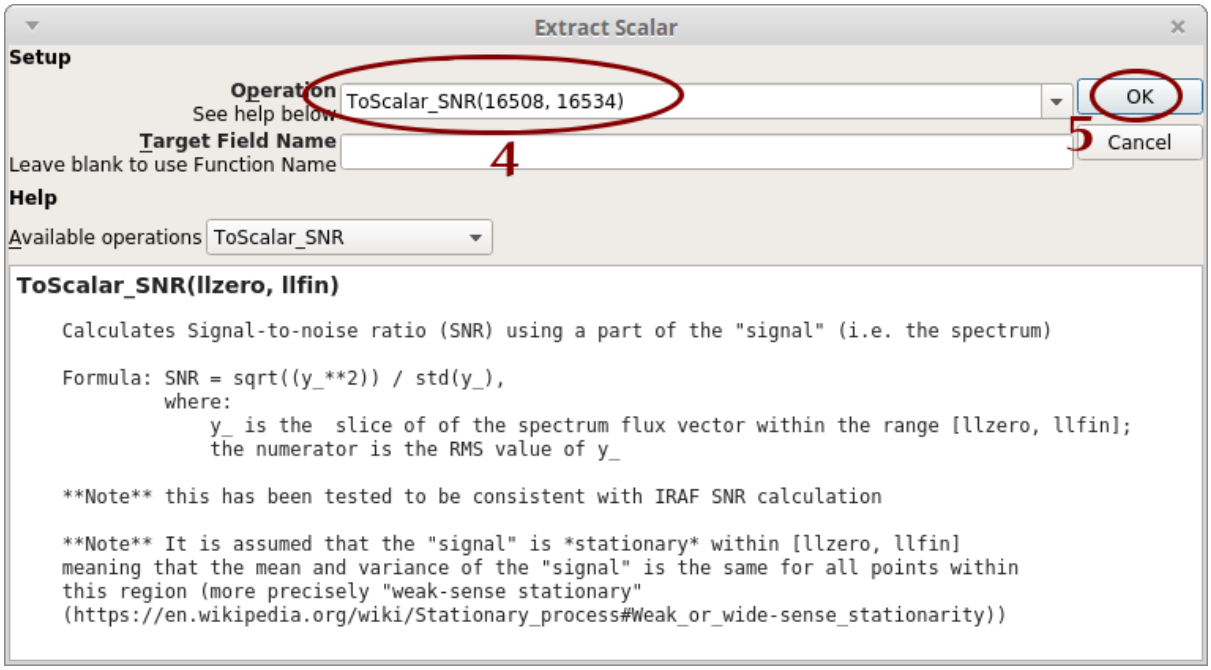
1. Select all the spectra: click inside the table, then press **Ctrl+A**



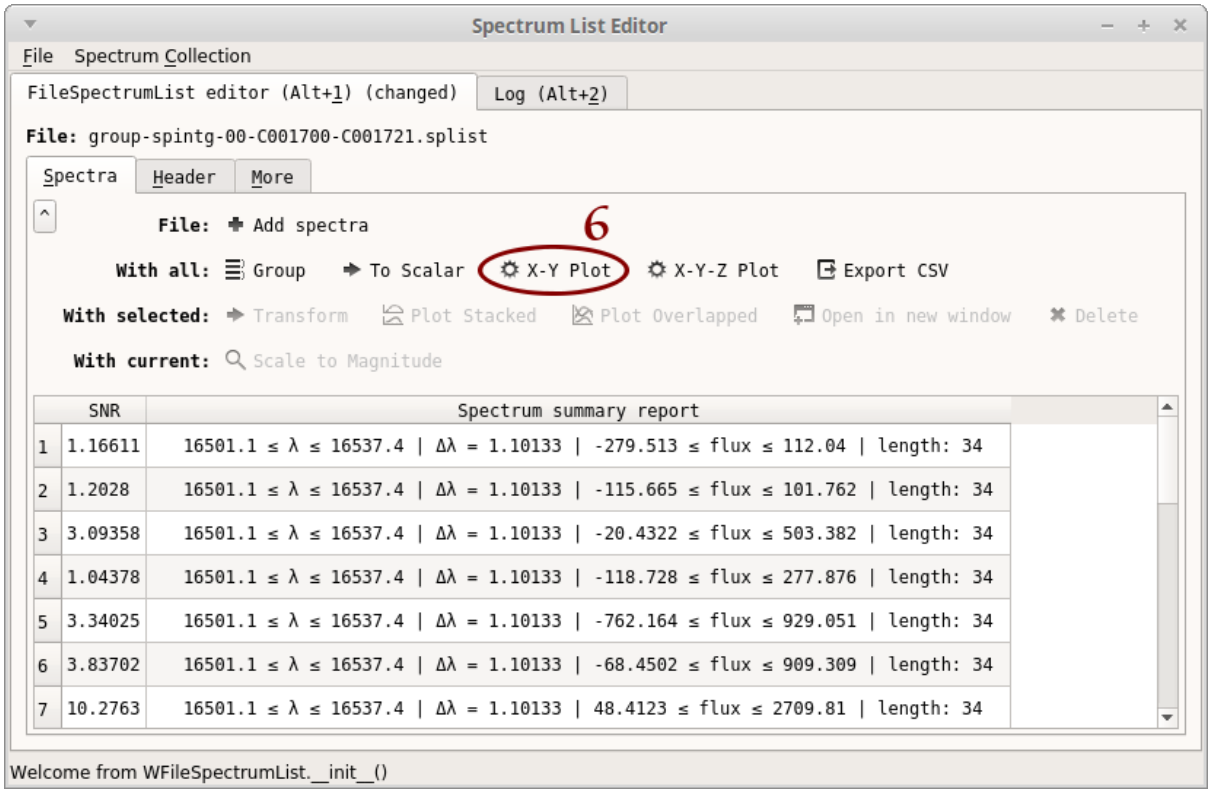
2. Click on “Plot Overlapped”. A plot window opens. From this plot, we can see that the region 16508-16534 seems to be free of atmospheric contamination. You may close the plot window



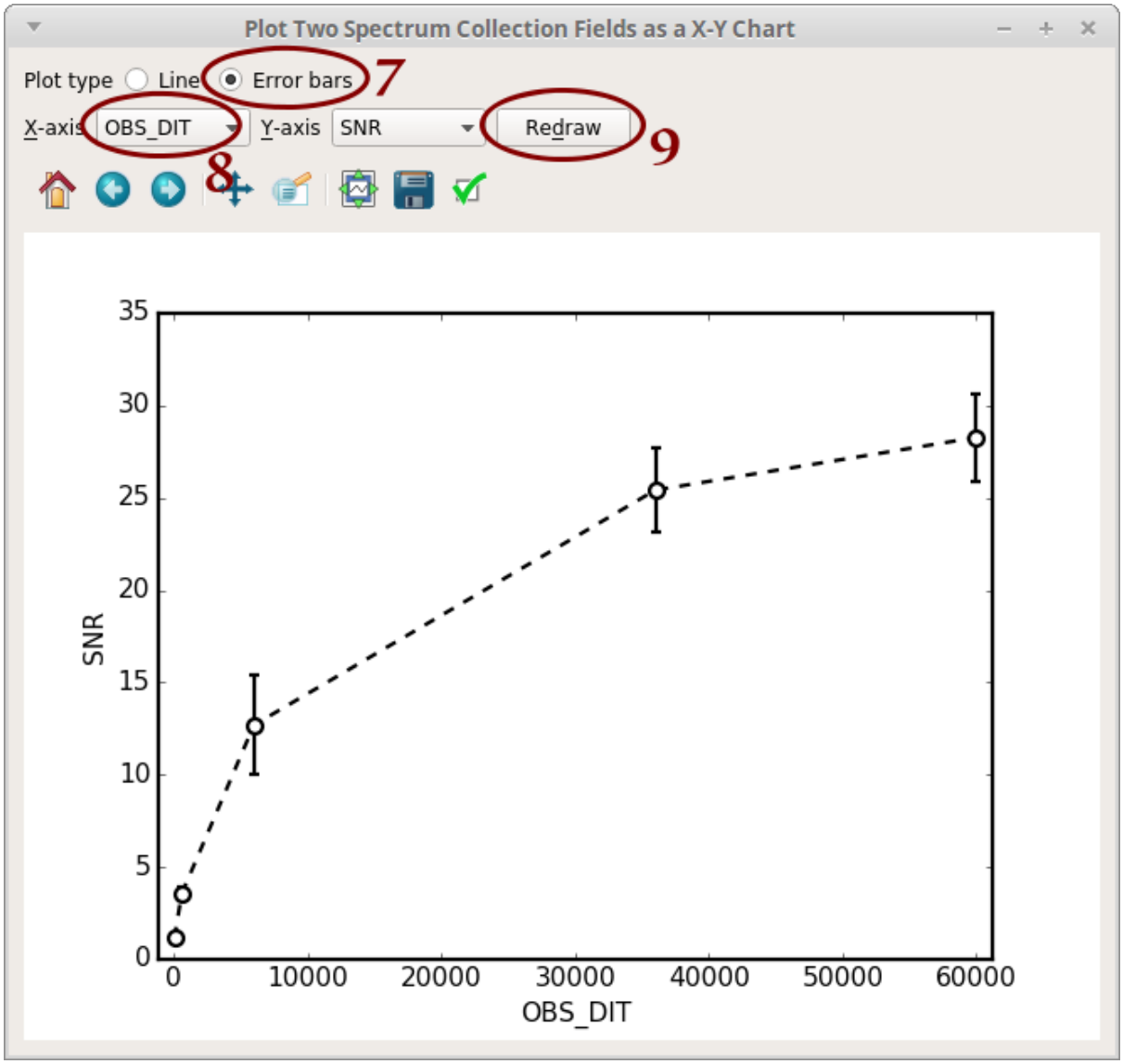
3. Click on “To Scalar”. Another window opens
4. Type “ToScalar_SNR(16508, 16534)”
5. Click on “OK”



6. Notice that a new column “SNR” appear in the table. Click on “X-Y Plot”



7. Select “Error bars”
8. Select “OBS_DIT”
9. Click on “Redraw”


Plot Two Spectrum Collection Fields as a X-Y Chart
Plot type
Line
Error bars
7
X-axis
OBS_DIT
Y-axis
SNR
Redraw
9
8
35
30
25
20
15
10
5
0
SNR
0
10000
20000
30000
40000
50000
60000
OBS_DIT

CHAPTER

FOUR

SPECTROMETRE MODES & SPECTRAL LINES OF INTEREST AT REDSHIFT

`wavelength-chart.py`

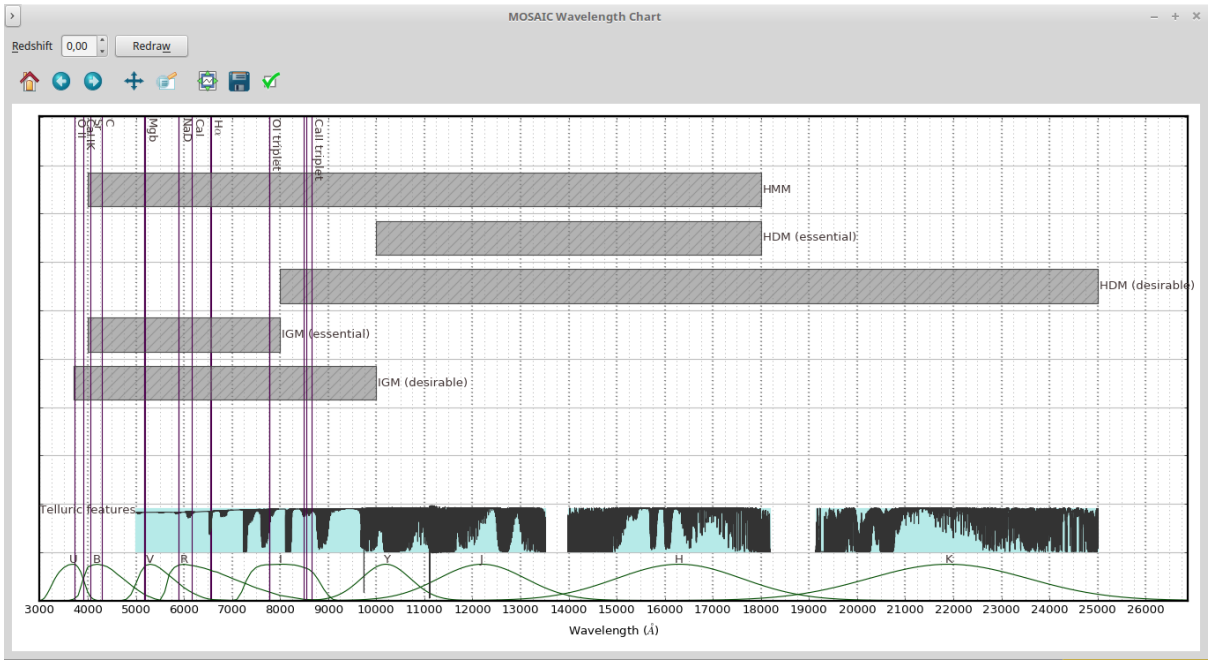


Figure – Lines with zero redshift

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:

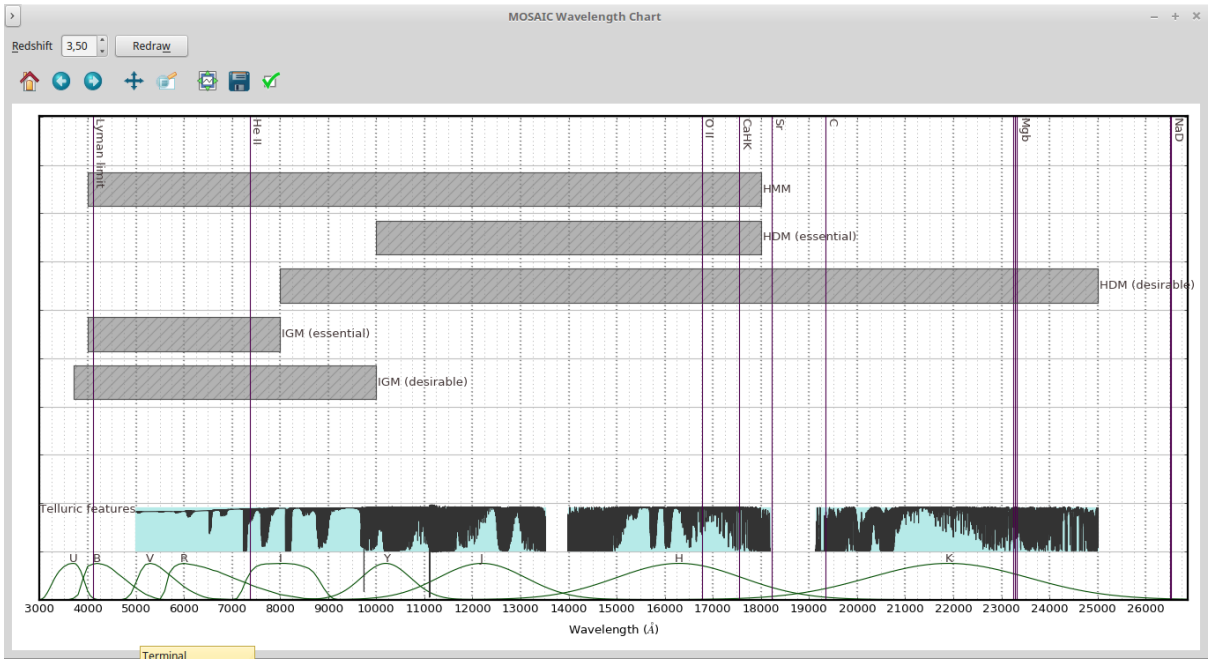

MOSAIC Wavelength Chart
Redshift
3,50
Redraw
Lyman limit
He II
O II
CaHK
Sr
G
Mgb
NaD
HMM
HDM (essential)
HDM (desirable)
IGM (essential)
IGM (desirable)
Telluric features
U
B
V
R
I
Y
J
H
K
3000 4000 5000 6000 7000 8000 9000 10000 11000 12000 13000 14000 15000 16000 17000 18000 19000 20000 21000 22000 23000 24000 25000 26000
Wavelength (Å)
Terminal

Figure – z=3.5

CHAPTER

FIVE

SCALE TO MAGNITUDE

Both `splisted.py` and `cubeed.py` have a “Scale to Magnitude” button that can be used to scale spectra to a desired magnitude in a given magnitude system (standard/AB/Vega) (Figure 5.1).

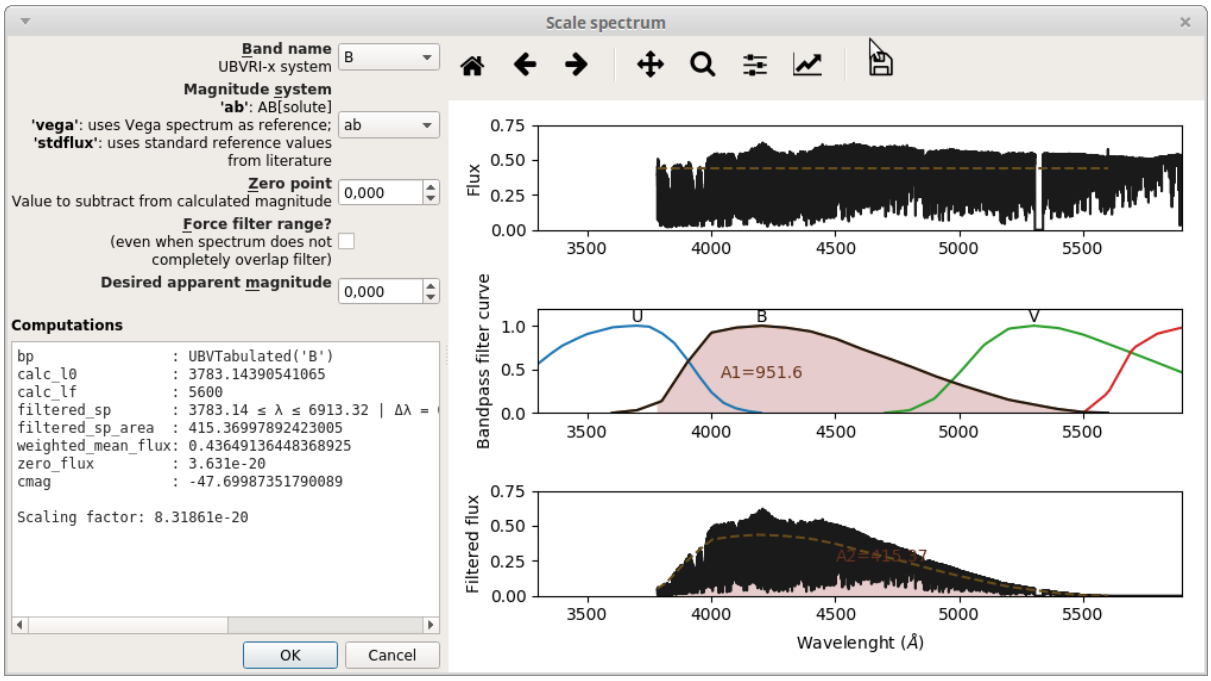


Figure 5.1: – “Scale to Magnitude” window

CHAPTER

SIX

INDEX OF APPLICATIONS (SCRIPTS)

This chapter is a reference to all scripts in project AOSSS

6.1 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 *aosss*

6.2 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 *aosss*

6.3 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 *aosss*

6.4 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 *aosss*

6.5 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 *aosss*

6.6 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 *aosss*

6.7 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 *aosss*

6.8 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 *aosss*

CHAPTER

SEVEN

PHOTOMETRY & COLORS API

7.1 Introduction

This section illustrates the API that was developed to solve photometry-related and color-conversion-related problems to compose the applications `cubeed.py` and `splisted.py`. Some usage examples of this API in further contexts are shown below.

7.2 Examples

7.2.1 Plot bandpass filter shapes

```
import aosss.physics as ph
import matplotlib.pyplot as plt
import numpy as np


l0, lf = 3000, 250000
x =  np.logspace(np.log10(l0), np.log10(lf), 1000, base=10.)

ax = plt.subplot(211)
for name in "UBVRI":
    bp = ph.UBVTabulated(name)
    plt.semilogx(x, bp.ufunc()(x), label=name)
plt.xlim([l0, lf])
plt.title("Tabulated")

plt.subplot(212, sharex=ax)
for name in "UBVRIYJHKLMNQ":
    bp = ph.UBVParametric(name)
    plt.semilogx(x, bp.ufunc()(x), label=name)
plt.xlim([l0, lf])
plt.xlabel("Wavelength ($\AA$)")
plt.title("Parametric")
l = plt.legend(loc='lower right')

plt.tight_layout()
plt.show()
```

7.2.2 Passing spectrum throught bandpass filter

```
import aosss.physics as ph
import matplotlib.pyplot as plt
import numpy as np
```

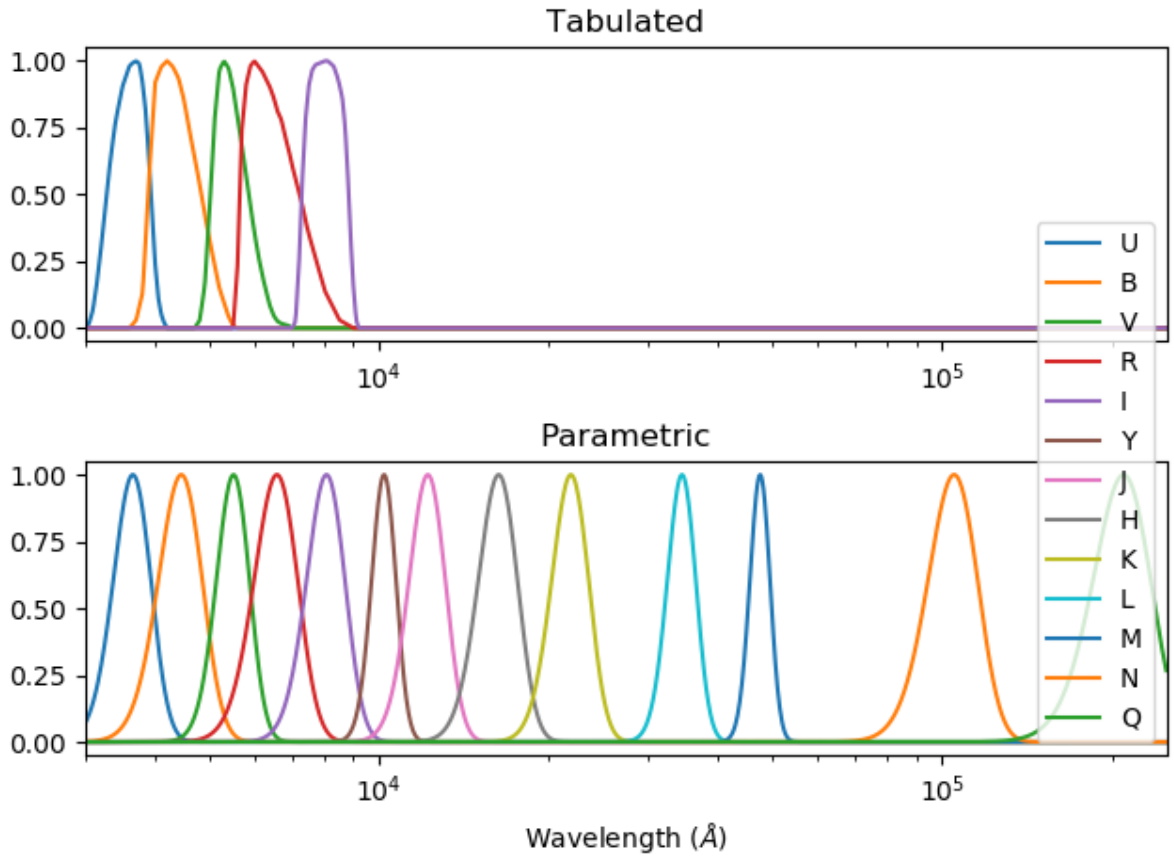


Figure 7.1: – Bandpass filter shapes in both tabulated and parametric formats.

```
BAND_NAME = "B"
STYLE = {"color": (0, 0, 0), "lw": 2}
sp = ph.get_vega_spectrum()

bp = ph.UBVTabulated(BAND_NAME)
filtered = sp*bp
x_band = sp.x[np.logical_and(sp.x >= bp.l0, sp.x <= bp.lf)]

ax = plt.subplot(311)
plt.plot(sp.x, sp.y, **STYLE)
plt.title("Source Spectrum (Vega)")

plt.subplot(312, sharex=ax)
plt.plot(x_band, bp.ufunc()(x_band), **STYLE)
plt.ylim([0, 1.05])
plt.title("Bandpass Filter (%s)" % BAND_NAME)

plt.subplot(313, sharex=ax)
plt.plot(filtered.x, filtered.y, **STYLE)
plt.title("Filtered Spectrum")
plt.xlabel("Wavelength ($\AA$)")

ax.set_xlim([bp.l0-100, bp.lf+100])
plt.tight_layout()
plt.show()
```

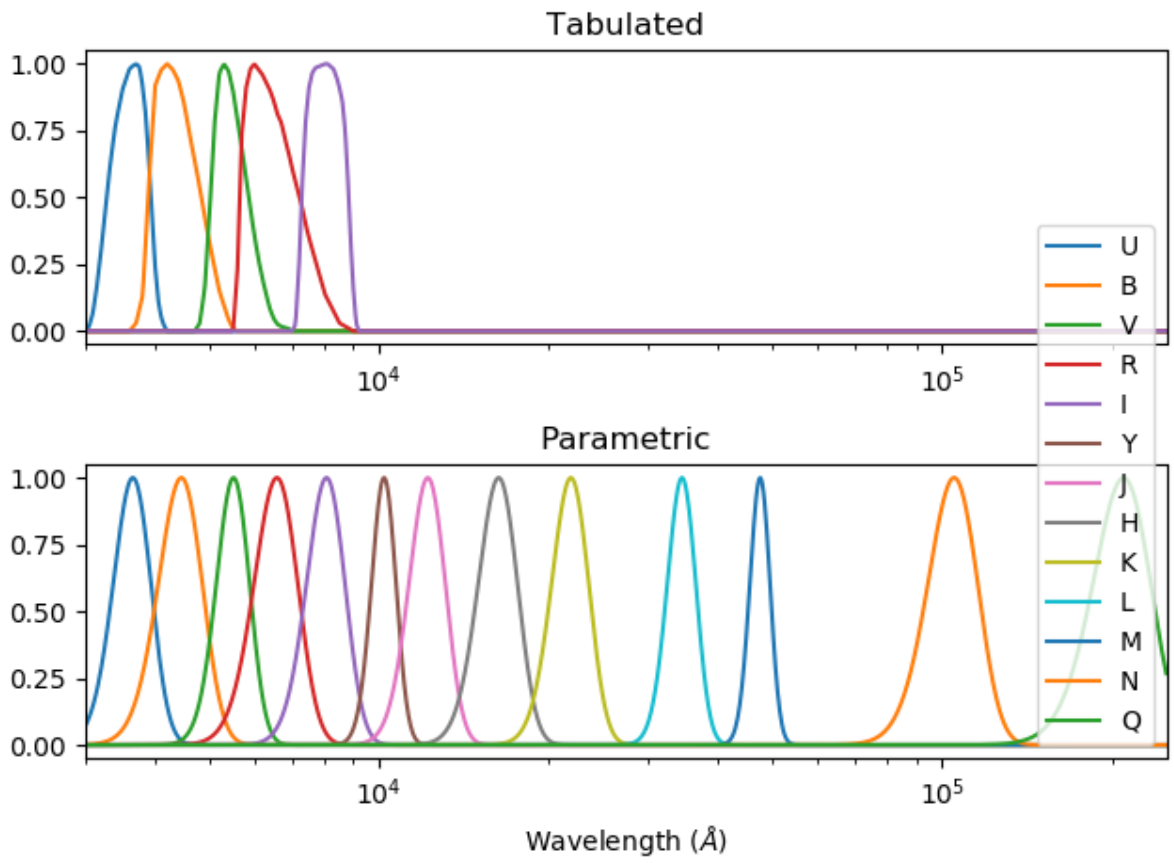


Figure 7.2: – Original spectrum, bandpass filter, and filtered spectrum.

7.2.3 Magnitude of spectrum for several bands and systems

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
```

7.2.4 Convert spectra to RGB colors

The following code plots blackbody spectra using color calculated from their respective spectra. This procedure can be applied to any spectrum.

```
# Plots blackbody curves (normalized to max=1.0) for red, yellow and blue stars;
# Calculates colors for these stars

import matplotlib.pyplot as plt
import numpy as np
import aosss.physics as ph
import f311

# color, temperature (K)
stars = [("blue", 10000),
         ("yellow", 5700),
         ("red", 4000),
         ("very red", 3000),
         ]

h = 6.626e-34
c = 3.0e+8
k = 1.38e-23

def planck(wav, T):
    """
    Calculates blackbody curve. wav in angstrom, T in kelvin. Returns intensity vector

    Adapted from https://stackoverflow.com/questions/22417484/plancks-formula-for-blackbody-spectrum
    """

    wav_ = wav*1e-10 # converts to m

    a = 2.0*h*c**2
    b = h*c/(wav_ * k * T)
```

```
    intensity = a/ ((wav_ ** 5) * (np.exp(b) - 1.0))
    return intensity

# x-axis wavelength in angstrom
wavelengths = np.linspace(10., 30000., 1000)

plt.style.use("dark_background")
for name, temperature in stars:
    flux = planck(wavelengths, temperature)
    flux /= np.max(flux)
    color = ph.spectrum_to_rgb(f311.Spectrum(wavelengths, flux))
    plt.plot(wavelengths, flux, color=color, label="{} star ({} K)".format(name, temperature))

plt.legend(loc=0)
plt.xlabel("Wavelength (angstrom)")
plt.ylabel("Flux (a.u.)")
plt.title("Normalized blackbody curves")
plt.tight_layout()
plt.show()
```

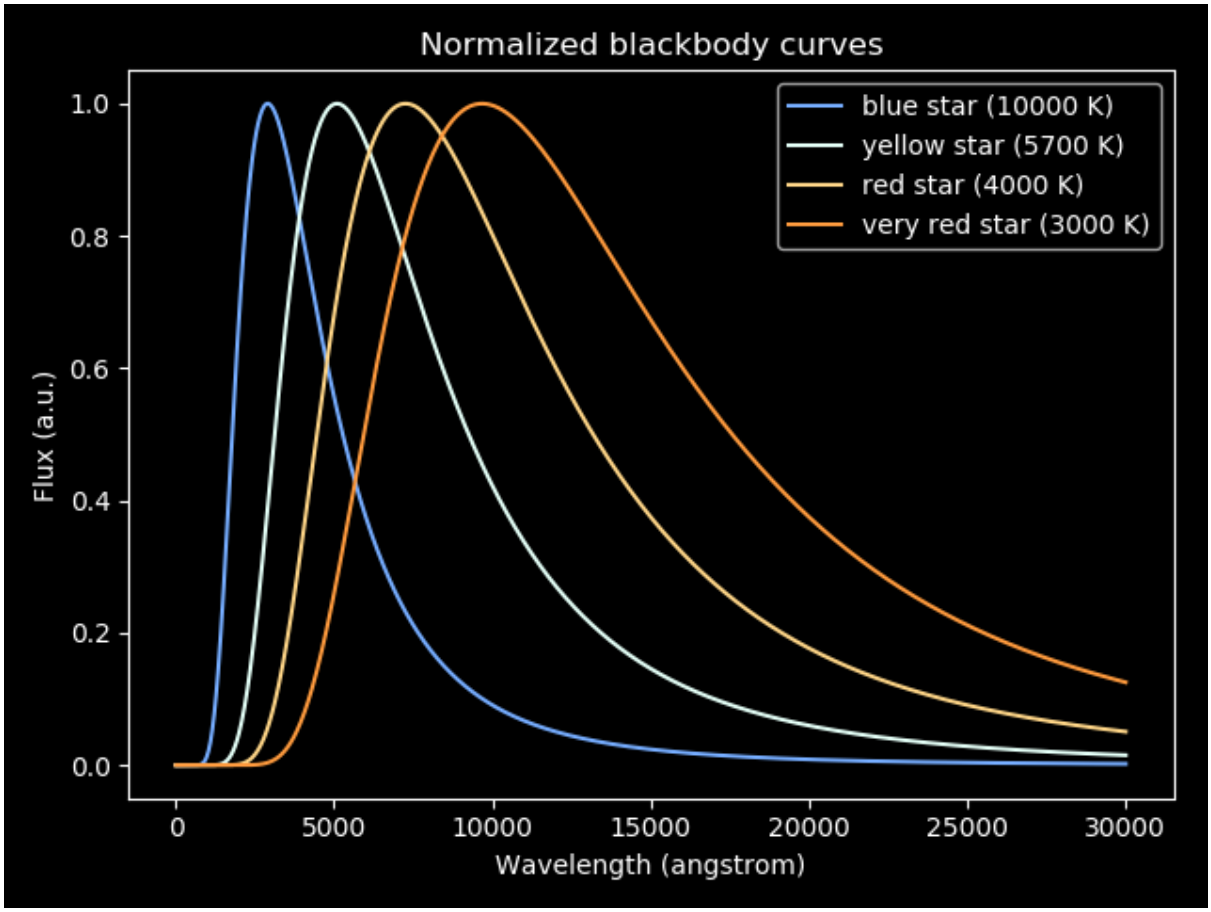


Figure 7.3: – Blackbody spectra painted with colors calculated from the spectra themselves.