

# User Manual for Data Reduction of the EXOhSPEC Instrument

## Working version: V0.3.2

Ronny Errmann, Neil Cook  
Physics, Astronomy and Maths, University of Hertfordshire

December, 2018

## Contents

<b>1</b>	<b>Changelog</b>	<b>1</b>
1.1	v0.3.3	1
<b>2</b>	<b>Introduction</b>	<b>2</b>
<b>3</b>	<b>Installation and dependencies</b>	<b>3</b>
3.1	With Anaconda and python 2.7	3
3.1.1	Additional modules	3
3.2	Necessary files	4
3.3	General information about the way the pipeline works	5
3.3.1	Handling parameters	5
3.3.2	Finding further information	5
<b>4</b>	<b>Running the code on a new set of data (e.g. a new night)</b>	<b>6</b>
4.1	Steps performed by the pipeline	6
4.2	Necessary CCD images	7
4.3	Steps to run the code on a new set of data	8
4.4	Possible parameters for the CCD processing	8
4.5	Assigning the observed data and calibration data to the pipeline	10
4.6	Necessary information to calculate the barycentric correction	11
4.6.1	get_ut1_offset: requested mjd is beyond end of IERS file	12
4.6.2	Creating a list of objects	12
4.7	Format of the extracted files	12
4.8	Results from the pipeline	14
<b>5</b>	<b>Setting up the pipeline</b>	<b>20</b>
5.1	Creating a new wavelength solution	20
5.2	Setting up a new account or a new camera	21
5.3	Create a bad pixel mask	22
<b>6</b>	<b>Post-extraction analysis</b>	<b>23</b>
6.1	Plotting the data	23

<b>7 Extracting data from other spectrographs</b>	<b>24</b>
7.1 Extracting HARPS data . . . . .	24
7.1.1 Getting the data . . . . .	24
7.1.2 Preparing for data extraction . . . . .	25
7.1.3 HARPS data reduced by the CERES pipeline . . . . .	25
7.1.4 HARPS data reduced by the ESO Phase 3 pipeline . . . . .	26
7.1.5 Analysing the reduced HARPS data with TERRA . . . . .	27
7.2 Extracting MRES data . . . . .	28
<b>A Module dependencies if not using Anaconda</b>	<b>29</b>
<b>A Short introduction to reduction of CCD images</b>	<b>30</b>

# 1 Changelog

## 1.1 v0.3.3

- Added parameter `raw_data_timezone_cor` to `conf.txt`.

## 1.2 v0.3.4

- Added parameter `object_file` to `conf.txt`.
- More information are stored in the header of the different output files.
- Barycentric velocity calculation and barycentric JD were improved/added.

## 2 Introduction

This program is designed to take a fits image file (i.e. a CCD image) and run data reduction steps, extract out orders from an Echelle spectrograph (regardless of separation and curvature, as long as orders are distinguishable from one-another), apply the wavelength correction, measure the radial velocity, and perform further calibration steps.

## 3 Installation and dependencies

This program was written to work with modules installed in Anaconda 2 and python 2.7. It is recommended to use an installation of anaconda 2 to use this program (If this is not possible, the whole module dependency is given in Appendix A).

Similar to the CERES pipeline, the SSEphem package ([http://www.cv.nrao.edu/~rfisher/Python/py\\_solar\\_system.html](http://www.cv.nrao.edu/~rfisher/Python/py_solar_system.html)) in connection with the SOFA (<http://www.iausofa.org/>) functions are used.

### 3.1 With Anaconda and python 2.7

Anaconda can be downloaded from <https://www.anaconda.com/download/>. Please select the version for python 2.7. After download it can be installed by running

```
sh Downloads/Anaconda2-5.0.1-Linux-x86_64.sh
```

(no superuser permissions are needed, the version might need to be changed).

After the installation and opening a new terminal, python should point to the one in the Anaconda installation. It can be checked by running `which python`. If the result doesn't point to your Anaconda installation, but somewhere else (e.g. `/usr/bin/python`), an extra entry into `.bashrc` or `.tcshrc` needs to be added. If you use `bash` add the following line (replace `/home/exohspec` by the path you used for the installation to your `.bashrc`).

```
export PATH="/home/exohspec/anaconda2/bin:$PATH"
```

If you use `csh` add the following line (replace `/home/exohspec` by the path you used for the installation to your `.tcshrc`

```
setenv PATH /home/exohspec/anaconda2/bin\:$PATH
```

After making the change and opening a new terminal, python should point to the file in the Anaconda installation (`which python`).

#### 3.1.1 Additional modules

Once Anaconda is installed tow other modules are needed (`tqdm`, `gatspy`). If Anaconda is installed correctly then `pip` can be used to install this module

```
pip install tqdm
pip install gatspy
pip install ephem
pip install jplephem
pip install de423
```

To use the cross-correlation based on the CERES pipeline the package `pyfits` is necessary:

```
conda install gsl=2.2.1
pip install pyfits
```

If the installation of `pyfits` fails, the following two commands can be tried to install it under `conda`:

```
conda install -c cefca pyfits  
conda install -c sherpa pyfits
```

This might downgrade numpy, which will cause problems. Therefore, after installing pyfits, numpy needs to be upgraded again:

```
pip install numpy --upgrade
```

Updating anaconda might also solve it:

```
conda update --prefix /home/exohspec/anaconda2 anaconda
```

## 3.2 Necessary files

The Necessary python and configuration files can be loaded from github (adjust the version to the latest version by checking <https://github.com/ronnyerrmann/exohspec/releases>):

```
wget https://github.com/ronnyerrmann/exohspec/archive/v0.3.2.tar.gz  
tar -xzvf v*.tar.gz
```

The following files should be available now in the program folder:

**prepare\_file\_list.py** Script to automatically assign the fits files into the corresponding calibration steps.

**reduction\_day.py** Script which needs to be run on each new set of data. It creates the data, which is necessary in order to extract the scientific spectra. Afterwards it extracts the Echelle spectra of the science images.

**procedures.py** Contains all the procedures for the data analysis.

**tkcanvas.py** Contains the plotting routines to create the user interfaces (UI).

**plot\_img\_spec.py** This script controls plotting of CCD images, graphs, and the extracted spectra.

**remove\_orders.py** If the automatic process identified wrong orders, then this script allows the user to remove them using a GUI.

**create\_badpx\_mask.py** Script to create a bad pixel mask (in testing).

**SSEphem** Necessary folder to calculate barycentric velocity. (*It might be necessary to install the package to have the correct paths.*)

**ssephem\_update.py** Script to update the SSEphem package (See Section 4.6.1).

When running the scripts for the first time, python will create a new file called <filename>.pyc for some of the files (compiled script). No harm will be done in removing or keeping the files.

### 3.3 General information about the way the pipeline works

#### 3.3.1 Handling parameters

1. The script has some hard coded values in the \*.py files (normally at the beginning of a procedure). These values can be changed for testing, but usually do not need any changes.
2. The pipeline reads the parameters from a configuration file (standard: `conf.txt`), given at the beginning of the python scripts. Some of the parameters in the configuration file need adjustment on a regular basis (see Section 4.3).
3. Furthermore, the pipeline reads the configuration file given in parameter `configfile_fitsfiles` (standard: `fits_conf.txt`), which is automatically created by the pipeline when running `prepare_file_list.py`.
4. The parameters which are set in the configuration file(s) can be overwritten with a command line input when a python script is started (e.g. `python bla.py argument1=valueX argument2=valueY`).
5. Some parameters can be overwritten during the run time of the script by user input if the GUI is enabled.

#### 3.3.2 Finding further information

- The pipeline logs the execution of procedures and necessary information in a logfile (standard: `logfile`). Most of this information is also printed into the terminal window.
- All adjustable parameters are logged at the beginning and at the end of the execution of a python script into a logfile (standard: `logfile_params`).
- Images with the results of some steps can be found in a logging subfolder (standard: `logging`). Some of these images are shown in Figures 1 to 5.
- Each parameter is explained in detail in the configuration file.
- The input and output parameters of the individual procedures are explained in the file `procedures.py`.

## 4 Running the code on a new set of data (e.g. a new night)

### 4.1 Steps performed by the pipeline

The following steps will be performed on a new set of data:

1. Creating the following reduced and combined files:

**trace1** File in which the science traces can be determined (standard: 5x white light flats, best without arc lamp). This file is also used to create a map of the background flux.

**trace2** File in which the wavelength calibration traces can be determined (standard: 5x ThAr alone (future development: white light flats)).

**arc\_l** : Long arc exposure to create a good wavelength solution over the whole chip (standard: 5x 100 s ThAr or UNe).

**arc\_s** : Short exposure in order to find the center of the saturated lines in arc\_l (standard: 5x 10 s ThAr or UNe taken between the arc\_l images).

**flatarc** : File which contains the blaze correction in the science fiber (standard: 5x files with the white light flats in science fiber).

2. Determining the shift of the science traces compared to the previous solution (e.g. previous day). If the instrument was not touched, the shift should be small and constant (→ Fig. 1). If the file for the previous solution does not exist, or if a big deviation to the previous solution has been found, then another step will be executed:

- 2a. Finding the traces of all science orders. First in a heavily binned image (20,5) to find the traces, and then in a slightly binned image (only few pixel in dispersion direction) where the position is redefined.
3. (for bifurcated fiber spectrographs) Finding the traces of the calibration orders by searching for the shift between each order in the trace1 and trace2 file (→ Fig. 2).
4. Create the wavelength solution for the night (→ Fig. 3 and 5). This is based on the solution of the previous data set, which is adjusted according to the given parameters.
5. Extract the flat and normalise it. This is used for the blaze correction.
6. (for single fiber spectrographs) Measure the instrumental drift in emission line spectra to calibrate the wavelength solution.
7. Extract the science spectra. This includes the following steps for each image:
  - a) Data reduction of the CCD frame (if set up).
  - b) Finding the shift between this frame and the **sflat** file.
  - c) Extraction of the apertures for the science fiber.

- d) (for bifurcated fiber spectrographs) Extraction of the apertures for the calibration fiber.
- e) Finding the shift between the emission lines in the calibration spectra and the lines used for the wavelength solution (for bifurcated fiber spectrographs) or interpolating the measured drift from step 6. Calculating the wavelength for each pixel in the extracted spectrum.
- f) Creating the flat corrected spectrum.
- g) Creating the spectrum with normalised continuum.
- h) Perform the radial velocity analysis (cross-correlation with template spectrum, work in progress).
- i) Perform some general measurements which are stored in the header.
- j) Write the file with all extracted data into the folder given in the parameter `path_extraction` (standard: `extracted`).
- k) Write subsets of data into different files to keep compatible with other software.

If the result file of any of the above steps already exist in the folder in which the code is run, then the existing file is read instead of performing the step again in order to save computational time. If a step needs to run again, the according result file needs to be deleted.

## 4.2 Necessary CCD images

To get the best results, the following data should be taken (all filenames are case-insensitive). **No spaces are allowed in the file names.**

- *true Flat (at least 11 files of the evenly illuminated CCD)*. This has tested to improve the data quality when using a camera with small full well depth. The filename should contain **rflat** for automatic processing.
- Flat (11x, white light source through the science fiber), with the filename containing **sflat** or **whli**. The calibration fiber should be dark for this data.
- Arc (5x, calibration lamp through the calibration fiber, alternating a short (e.g. 5 s) and a long (e.g. 120 s) exposure time). The filename should start with **arc**, **ThAr**, or **UNe** and the science fiber should be dark for this data.
- Bias (11x) and/or Darks (11x, for the exposure time of the true Flat and Flat)
- Filenames science images: the optimum is to use “\_” for object name (as in reference file `object_file`, see Chapter [4.6.2](#)). Use “-” to separate exposure time, number of series and other information.

### 4.3 Steps to run the code on a new set of data

1. Create and enter a new folder. This folder will be used for the reduced and extracted data.
2. Copy the configuration file (`conf.txt`) into the folder. The following parameters might need to be changed in the file `conf.txt`:

**GUI** : If set to true, this allows manipulation of some parameters in a graphical user interface (GUI) during the runtime of the script. (not tested recently)

**raw\_data\_path** : Folder to the raw data.

**original\_master\_traces\_filename** : Path to the traces of the science orders from the previous day. This file will be the base for finding the traces in the current night ([Step 2] as described in Section 4.1). Leave empty if the science orders should be searched without taking the previous solution into account (e.g. after the setup of the spectrograph has changed).

**original\_master\_arc\_solution\_filename** : Path to the previous wavelength solution. This solution will be used as base for the wavelength solution in the current night. In case of a new setup see Section 5.1. In case no wavelength solution is required, set it to `pseudo`.

**<type>.calibs.create\_g** : Change the calibrations, which are applied to the CCD images before further processing or extraction for the different file types (e.g. bias, dark, flat, sflat, arc, flatarc, extract, wavelengthcal). A list of possible calibration options is given in Section 4.4 below.

3. Run the python script:

```
python <path to scripts>/prepare_file_list.py .
```

See Section 4.5 below for more information.

4. Run the python script:

```
python <path to scripts>/reduction_day.py .
```

5. Check the out in the `logfile`, the images in the logging path, or in results in the extracted files.

After checking the results (see Section 4.8), one might want to remove wrongly identified apertures. If a trace is located close to the borders this can lead to wrongly traced orders, which can't be handled by the code automatically due to its big flexibility. The orders can be removed in a graphical user interface by running `remove_orders.py`. The old files will be moved into a sub-folder and the depending steps of `reduction_day.py` will run again.

### 4.4 Possible parameters for the CCD processing

The list below shows all standard CCD calibration options available in the pipeline. The corrections are done on a pixel-by-pixel basis. The steps will be executed in the same order as given here.

**subframe** : The subframe as given in the parameter **subframe** will be applied, only a section of the CCD will be used.

**badpx\_mask** : Apply the bad pixel mask, given in parameter **badpx\_mask\_filename**. If the file doesn't exist, all pixels are assumed to be good.

**bias** : Apply a master bias, which will be created from the files given in the parameter **bias\_rawfiles** by using the settings given in the parameter **bias\_calibs\_create**. These two parameters are created automatically by the script **prepare\_file\_list.py** if bias frames exist in the raw data path.

**dark** : Apply a master dark. A dark with the same exposure time as read from the fits header will be applied. The necessary parameters are created automatically by the script **prepare\_file\_list.py**.

**rflat** : Apply a master true flat. The necessary parameters are created automatically by the script **prepare\_file\_list.py**.

**localbackground** : Applies background correction by fitting a 2d polynomial against the current reduced image, excluding the area of the science and calibration traces. The fitted 2d-image is then subtracted from the (reduced) image.

The following parameters change the way, several CCD images are combined. In the pipeline always the individual frames are corrected before frames are combined.

**normalise** : Before combining the files the images are normalised by their median flux.

**combine\_sum** : Normally, files are combined using a pixel by pixel median. With this keyword the files will summed up. This parameter will be overwritten by **combine\_mean**. When summing up several images, the value of pixels might extend **max\_good\_value**.

**combine\_mean** : Normally, files are combined using a median. With this keyword a pixel by pixel average will be used.

The calibration options can be altered, as long as they contain the unique string in the option name. For example an option **subframe\_1** can be used, in this case the parameter **subframe\_1** needs to be defined in one of the calibration files. If a different bias, dark, or flat should be used, the following parameters need to be included in one of the configuration files (for the example of *darkfixed*):

- **darkfixed\_rawfiles**
- **darkfixed\_calibs\_create**
- **master\_darkfixed\_filename**.

## 4.5 Assigning the observed data and calibration data to the pipeline

The script `prepare_file_list.py` reads all files in the folder (and subfolders) given in parameter `raw_data_path`. The file name and header information are used to determine the type of file and if it can be used for calibration. The header parameters are thereby defined by the following parameters in the configuration file:

`raw_data_imtyp_keyword`: Header keyword if the image type (standard: IMAGETYP),

`raw_data_imtyp_bias`: Value of the for the header keyword  
`raw_data_imtyp_keyword` for bias frames (standard: Bias Frame),

`raw_data_imtyp_dark`: Value of the for `raw_data_imtyp_keyword` for dark frames (standard: Dark Frame),

`raw_data_imtyp_flat`: Value of the for `raw_data_imtyp_keyword` for flat frames (standard: Flat Frame),

`raw_data_imtyp_trace1`: Value of the for `raw_data_imtyp_keyword` for frames to trace the science orders (for HARPS: LAMP,DARK,TUN),

`raw_data_imtyp_flatarc`: Value of the for `raw_data_imtyp_keyword` for frames with the blaze function (for HARPS: LAMP,LAMP,TUN),

`raw_data_imtyp_trace2`: Value of the for `raw_data_imtyp_keyword` for frames to trace the calibration orders (for HARPS: WAVE,WAVE,THAR2),

`raw_data_exptime_keyword`: Header keyword for the exposure (standard: EXPTIME),

`raw_data_dateobs_keyword`: Header keyword observing date and time (in UTC). The format needs to be YYYY-MM-DDTHH:MM:SS (e.g. 2018-02-28T23:34:01) (standard: DATE-OBS). Other formats can be defined in the procedure `get_obsdate`.

The file type and definition of the fibers is done by using the filename and header information. The assignment is done in the order given in Table 1. The result of this assignment is stored in a text file, which is shown to the user. The file can be edited (these information won't be overwritten if the script is re-executed). The following information is stored for each file in the raw data path (tab-separated):

- Filename
- Type of light for the science fiber
- Type of light for the calibration fiber
- Exposure time in seconds
- Observation time

Table 1: Assignment of the fibers using the header keywords as given in the parameters of the configuration and filename. The science and calibration fibers are denoted with 'fiber1' and 'fiber2', respectively. Later assignment overwrites earlier ones. The filename is case-insensitive. '-' means no change has been done.

condition	fiber1	fiber2
filename contains <b>bias</b>	bias	bias
filename contains <b>dark</b>	dark	dark
filename contains <b>flat</b> but not <b>sflat</b>	flat	flat
<b>raw_data_imtyp_keyword</b> equals <b>raw_data_imtyp_flat</b> and filename doesn't contain <b>sflat</b>	flat	flat
filename contains <b>arc</b>	-	wave
<b>raw_data_imtyp_keyword</b> equals <b>raw_data_imtyp_bias</b>	bias	bias
<b>raw_data_imtyp_keyword</b> equals <b>raw_data_imtyp_dark</b>	dark	dark
none of the above and the filename doesn't start with <b>arc</b>	science	-

- Flag 'w', if the Arc spectrum should be extracted (important for single fiber spectrographs). Flag 'e', if the spectrum should be extracted. This can be modified in order to allow different processing of the images before extraction. The following options are possible:

**e alone:** The processing as given in parameter `extract_calibs_create_g` will be assigned.  
**e <obj >:** The processing as given in parameter `extract<obj>_calibs_create_g` will be assigned. If this parameter doesn't exist, then parameter `extract_calibs_create_g` will be used.  
**ec <obj >:** The same as **e <obj >**, but instead of extracting each file individually, all files with `ec <obj >` will be combined into a file called `master_<obj>.fits` and then this file is extracted.

One raw file can be used for different extractions if several flags are combined with ',', e.g. `e,ecSunAll,ecSunCentroid`.

## 4.6 Necessary information to calculate the barycentric correction

In order to calculate the barycentric correction the time of the observation (or Julian Date), the pointing on the sky, and the position of the observatory on earth need to be known. These information can be given in different ways.

- The mid-exposure date and time is derived from the image header using the keys given in the parameter `raw_data_dateobs_keyword` and `raw_data_exptim_keyword` (half of the exposure time). The pipeline can handle different standard formats (YYYY-MM-DDThh:mm:ss). If necessary, further formats can be added in the procedure `get_obsdate`.
- The position of the observatory will be extracted from the following sources (using the first available source in the following list):

1. Reading latitude, longitude, and elevation from the header, using the header keys defined in the beginning of procedure `get_barycent_cor`: site\_keys, altitude\_keys, latitude\_keys, longitude\_keys (the first available entry of each list will be used, if necessary these lists can be extended).
  2. Using the site coordinates as given in the configuration file.
- The pointing of the telescope (RA and DEC) will be derived from the following sources (using the first available source in the following list):
    1. reading the object coordinates from a list of objects (see Chapter 4.6.2).
    2. Reading the object coordinates and epoch from the image header, using the header keys defined in the beginning of procedure `get_barycent_cor`: ra\_keys, dec\_keys, epoch\_keys (the first available entry of each list will be used, if necessary these lists can be extended).
    3. If the object name contains sun or moon, then the coordinates of the sun/moon are calculated for the mid of the observing time.

#### 4.6.1 `get_ut1_offset: requested mjd is beyond end of IERS file`

If the message `get_ut1_offset: requested mjd is beyond end of IERS file` appears, the data from Jet Propulsion Laboratory Development Ephemeris is out of date. It can be updated by running `ssephem_update.py`.

#### 4.6.2 Creating a list of objects

If the coordinates of the object are not stored in the fits header, then they can be given using a the file specified in parameter `object_file` (standard: `object_list.txt`). The file must contain one line per object. For each object the following tab-separated values need to be given:

1. Object name (the filename should start with the object name in order to be used)
2. RA (hh:mm:ss.ss or as one number in degrees)
3. DEC ( $\pm$ dd:mm:ss.ss or as one number in degrees)
4. PM RA (arcsec/year, can be 0 in the most cases)
5. PM DEC (arcsec/year, can be 0 in the most cases)
6. Epoch of the coordinates (only the number, e.g. 2000 or 2015.5)
7. Enable/Disable flag (1 for enabled)
8. Mask to use. At the moment G2, K2, or M2 are possible options. Can be empty
9. Velocity width to broaden the lines of the binary mask

#### 4.7 Format of the extracted files

Different types of extracted files are created. They are described in the following subsections.

### **raw-data filename**

The final fits file contains the original header and some additional information, for example the calibration steps which were applied and some information about the flux collected in the different apertures. The data in the file is stored in a 3D array in the form: data type, aperture, and pixel. The data types are similar to the ones created by the CERES pipeline and are the following:

0. 2D array with the wavelength for each aperture and pixel.
1. Extracted spectrum without any modification.
2. Measurement of the error the extracted spectrum (using the scatter of the residuals after fitting a 2d polynomial to the background areas (noise in the dark/bias))
3. Blaze corrected spectrum, calculated by dividing the extracted spectrum and the normalised flat spectrum.
4. Error of the flat corrected spectrum (residuals of a polynomial fitted to the blaze corrected spectrum).
5. Continuum normalised spectrum. The continuum is derived by fitting a polynomial to the flat corrected spectrum, using only areas of the spectrum where no lines are located.
6. Signal to noise ratio in the continuum, calculated from the residuals between continuum fit and measured continuum and the flux in the continuum.
7. Mask with good areas of the spectrum. The following values are used:
  - 1 good
  - 0 no data available
  - 0.1 saturated pixel in the extracted spectrum
  - 0.2 bad pixel in the extracted spectrum
8. Spectrum of the calibration fiber, e.g. of the emission line lamp.

### **raw-data filename + \_harps\_e2ds**

This file contains the extracted spectrum without any modification in the same form as the *e2ds* files created by the HARPS pipeline. The wavelength solution is stored in the header.

### **raw-data filename + \_lin**

The extracted spectrum is linearised using a wavelength step as given in parameter `wavelength_scale_resolution`. The wavelength spectrum is interpolated by using the weighted mean of the neighbouring data points with the wavelength difference as weights.

### **raw-data filename + \_lin.cont**

This file contains data in the same form as described in Section 4.7, only that the continuum corrected data was linearised.

### **inside single folder**

raw-data filename + \_extr Extracted spectrum with wavelength solution in IRAF format (wavelength solution without barycentric correction).

raw-data filename + \_blaze Blaze corrected spectrum with wavelength solution in IRAF format (wavelength solution without barycentric correction).

raw-data filename + \_wave 2D array with the wavelength for each aperture and pixel.

raw-data filename + \_weight Mask with good areas of the spectrum.

### **Object1/data/YYYY-MM-DDHHMMSS**

The data in here is used in order to measure RV with Terra (<https://drive.google.com/file/d/1xK-1YghFwpwdXG9b4IbryYRd102q7So/view>). The barycentric corrected wavelength solution and the continuum corrected spectrum (multiplied by the bad-pixel mask) are used. (work in progress)

## **4.8 Results from the pipeline**

For each of the steps given in Section 4.1 the following data will be created.

**Step 1:** The reduced and combined CCD images will be stored in the folder where the pipeline was run. The file name is `master_<type>.fits`, where `<type>` defines the different image types, e.g. bias, sflat, or arc.l.

**Step 2:** The trace of each scientific order is stored in the file given in parameter `master_traces_filename` (standard: `master_traces.fits`). This file contains one line for each order, normally starting with the reddest order, which is located on the left side of the CCD image. Each line contains the following information:

- Number of the aperture (starting at 0).
- Central pixel (in dispersion direction) for the polynomial fit along the center (Gaussian centre) of the trace.
- Parameters of a polynomial fit to the center of the trace (given in parameter `polynom_traces_apertures`), e.g. 5 values if the trace is fitted with a polynomial of order 5.
- Central pixel (in dispersion direction) for the polynomial fit along the left limit of the trace. The limit is determined from the value given in parameter `width_percentile`
- Parameters of a polynomial fit to the left limit of the trace (given in parameter `polynom_traces_apertures`).
- Central pixel (in dispersion direction) for the polynomial fit along the right limit of the trace. The limit is determined from the value given in parameter `width_percentile`

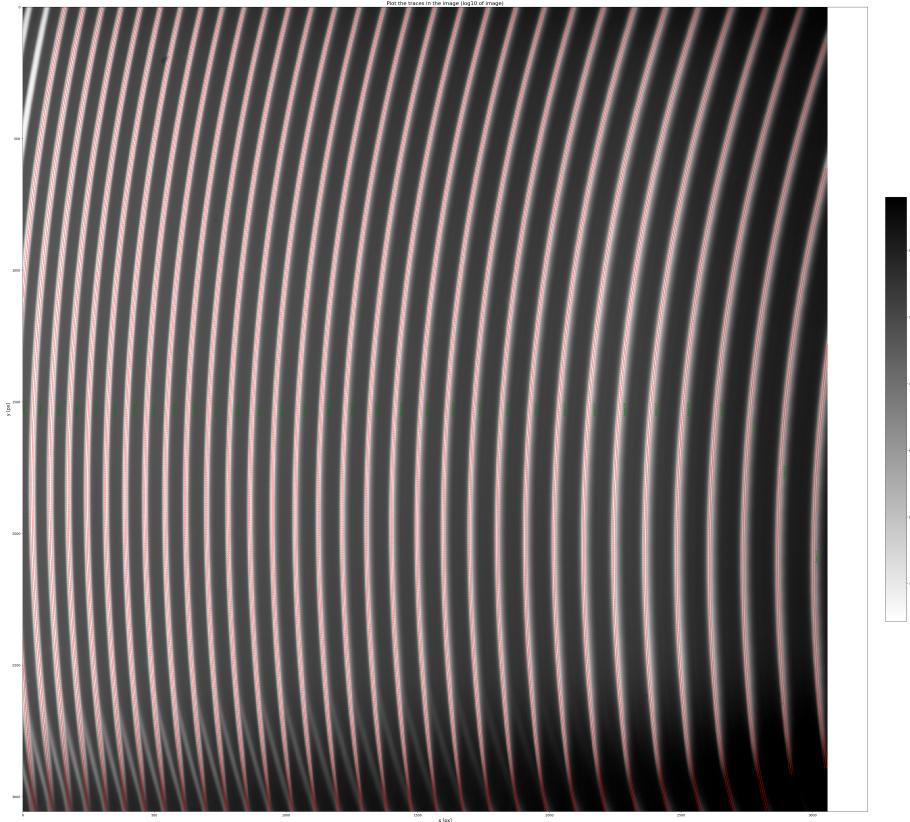


Figure 1: Reduced CCD image of a white light flat (log10 gray scale) with the marked traces of the identified scientific orders/apertures (red). The extraction width (determined from the width given in parameter `width_percentile` of the order multiplied with the value in parameter `extraction_width_multiplier`) is given in the dashed lines.

- Parameters of a polynomial fit to the right limit of the trace (given in parameter `polynom_traces_apertures`).
- The lowest and highest pixel in dispersion axis, for which the trace can be identified.
- The last three entries of each line define the width of the trace: left border (where the flux raises over the value given in parameter `width_percentile`), the right border (where the flux falls below `width_percentile`), and the Gaussian width.

The identification of the traces can be checked easily in the file given in parameter `logging_traces_im` (standard: `traces_in_master_trace1.png`). This file is located in the folder given in the parameter `logging_path` (standard: `logging`). An example is show in Fig. 1.

**Step 3:** The traces of the apertures of the calibration fiber are stored in the file given in parameter `master_tracesarc_filename` (standard: `master_tracesarc.fits`).

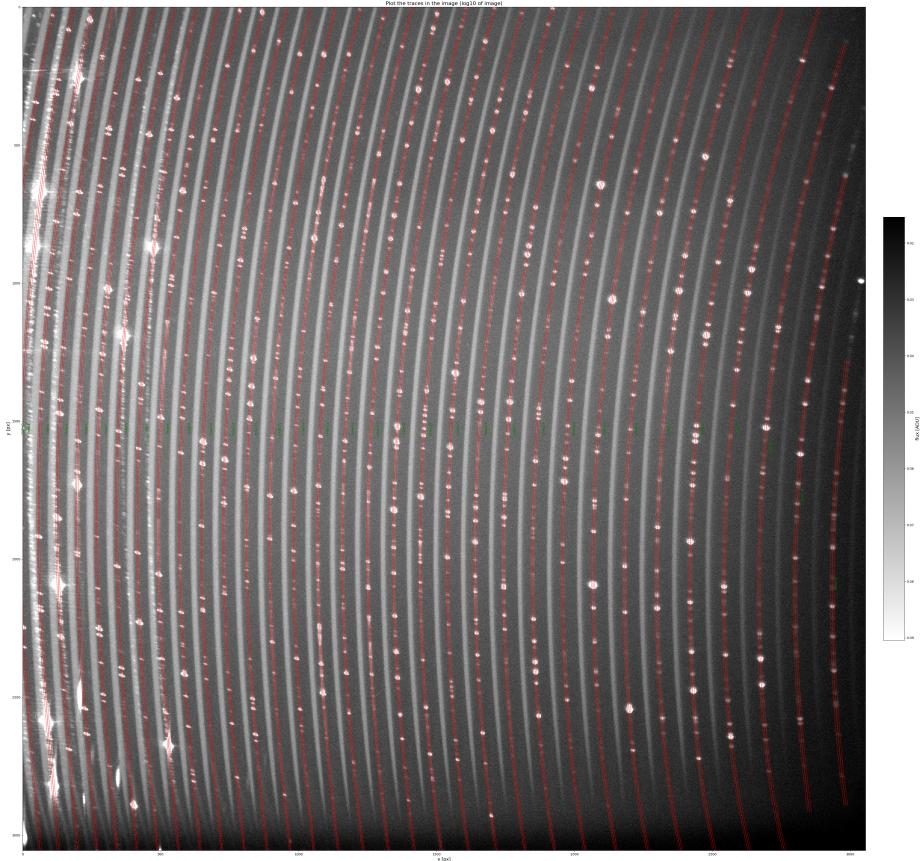


Figure 2: Reduced CCD image of a ThAr exposure (log10 gray scale) with the marked traces of the identified calibration orders/apertures (red). The extraction width (determined from the width given in parameter `width_percentile` of the order multiplied with the value in parameter `extraction_width_multiplier`) is given in the dashed lines.

This file contains the same information as the result of **Step 2**, only that the lowest order of the parameters of polynomial fit is shifted (the curvature of the traces is kept the same). The result can be checked easily in the file given in parameter `logging_arctraces_im` (standard: `arctraces_in_master_trace2.png`). An example is shown in Fig. 2.

**Step 4:** The wavelength solution is stored in the file given in parameter `master_wavelensolution_filename` (standard: `master_wavelength.fits`). The parameters for each order are stored in one line. For each order the following values are stored:

- Real order, as derived from the grating equation.
- Central pixel of the order (in dispersion direction).
- Parameters of a polynomial fit to the trace, e.g. 4 values if the dispersion axis is fitted with a polynomial of order 4.

- List of the wavelengths of all the identified reference lines in this order.

During the step to find the wavelength solution, the pipeline logs data similar to the following output:

```
Info: To match the most lines in the arc with the old wavelength solution, a shift
of -2 orders, a multiplier to the resolution of 0.994, a shift of -10 px, and a
shift of 0.0 px per order needs to be applied. 1046 lines were identified. The
deviation is 0.1197 Angstrom.
Info: used 1054 lines. The standard deviation of the residuals between the lines and the fit is 0.0344 Angstrom (the
Info: A 2d polynom fit with 5 orders in dispersion direction (along the traces) and 5 orders in cross-dispersion direc
ap cenwav minwav maxwav range Ang/ name numb gausswi gausswi min_ max_ range_reflin
px dth_avg dth_std reflin reflin _whole_order
0 7978.6 7894.7 8054.2 159.5 0.038 Ar I 2 1.78 0.58 7916.4 7948.2 98.1
0 7978.6 7894.7 8054.2 159.5 0.038 Th I 2 2.04 1.15 7937.7 8014.5 98.1
...
27 5800.1 5769.1 5854.3 85.2 0.027 Ar I 2 3.06 0.15 5802.1 5834.3 44.6
27 5800.1 5769.1 5854.3 85.2 0.027 Th I 6 2.31 0.41 5789.6 5832.4 44.6
-1 -1.0 -1.0 -1.0 -1.0 1.000 Ar I 148 2.07 0.61 5802.1 7948.2 2146.1
-1 -1.0 -1.0 -1.0 -1.0 1.000 Th I 376 2.04 0.66 5789.6 8014.5 2224.9
```

Thereby the table contains the following information:

**apert** Aperture of the trace, starting with 0 for the reddest aperture ( $\tilde{m}$ ). To get the real order the offset  $m_0$  is given in the text before (here: 72). The offset is determined using two different ways. First the data is compared to the grating equation  $\lambda \propto m_0 + \tilde{m}$ . In Practical this was done by searching for the smallest slope in the formula  $y = (m_0 + \tilde{m})\lambda_c$ , where  $\lambda_c$  is central wavelength of each order. The second value for the real order offset is determined from the shift towards the previous wavelength solution. Both values for the offset should be the same. Only the first value is saved in the file for the wavelength solution.

**cenwave** Central wavelength of the order, determines the zero point of the wavelength solution.

**minwave, maxwave, ranwave** Minimum and maximum wavelength, and wavelength range which is covered by the trace of this order.

**Ang/px** Resolution in  $\frac{\text{\AA}}{\text{px}}$  at the central wavelength.

**name** Type of the reference line. If different types of reference lines were found in the order then a output for each available type is created.

**number** Number of reference line for this type and order.

**gausswidth\_avg, gausswidth\_std** Average and standard deviation of the Gaussian width of the emission lines

**min\_refline, max\_refline** Minimum and maximum wavelength of the reference lines of this type and order

**range\_reflines\_whole\_order** Wavelength range which was covered by reference lines for this order (independent of type of the line)

The last lines give the information for all apertures. Columns, for which no useful information can be derived show the value -1.

**Step 5:** The normalised white light flat is stored in the file given in parameter `master_flat_spec_norm_filename` (standard: `master_flat_spec_norm.fits`). The extraction and data format is described in Section 4.7

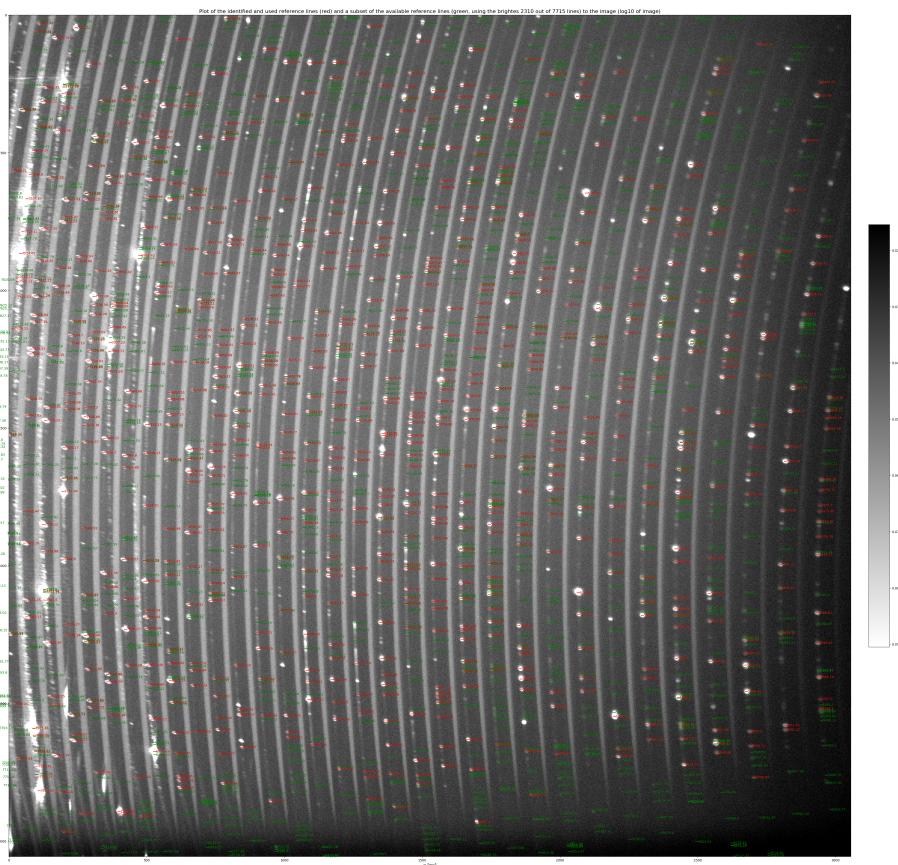


Figure 3: Reduced CCD image of a ThAr exposure (log10 gray scale) with the identified lines from the reference catalogue (red). Only this set of data was used to create the wavelength solution. The remaining lines of the reference catalogue, which weren't used for fitting the solution are shown in green. The data on an order by order basis is shown in Figure 4.

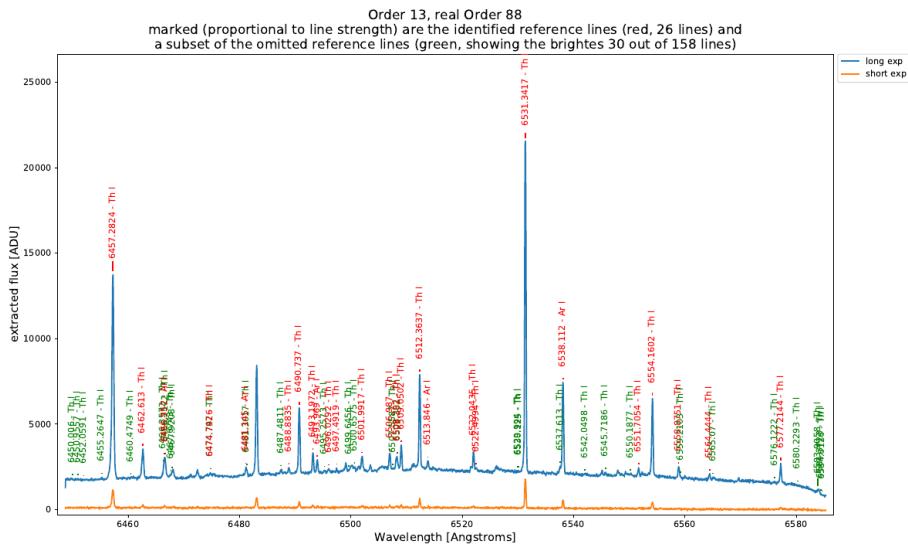


Figure 4: One page (aperture 13, order 88)real of the spectral atlas (file created from parameter **logging\_arc\_line\_identification\_spectrum**) created from the wavelength solution. Blue shows the emission lines in the long and orange the emission lines in the short exposure. The identified lines are marked in red, with the length of the marker being proportional to the intensity of the line as given in the **reference\_catalog** (length is reset for each order). Green shows a subset of the non-identified lines from the **reference\_catalog**, this means the green markers should normally be shorter than the red ones. Problems with an overcrowded **reference\_catalog** is visible at a few places (e.g. 6466 Å).

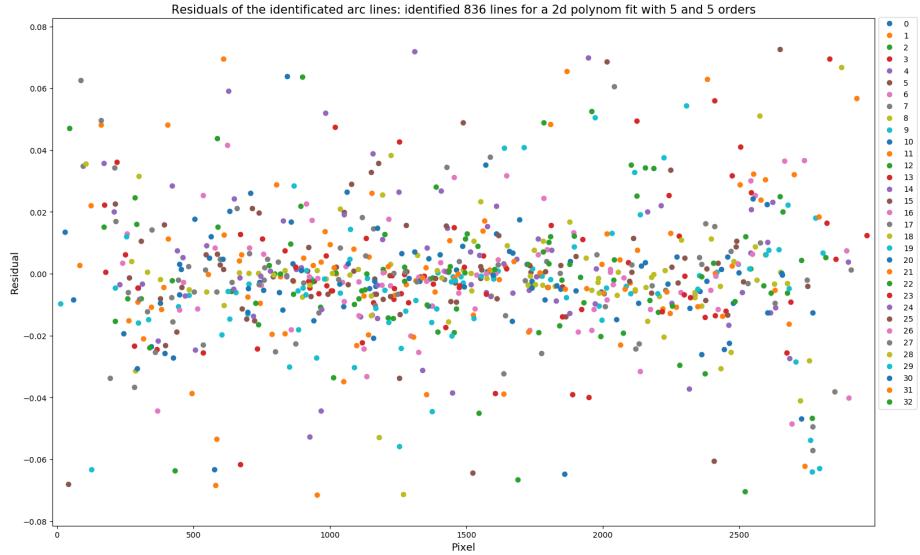


Figure 5: Residuals between the identified catalogue lines and the wavelength solution. Colored are the different apertures.

## 5 Seting up the pipeline

### 5.1 Creating a new wavelength solution

If no previous wavelength solution exist, then wavelengths and pixel need to correlated manually. In this case the parameter `original_master_arc_solution_filename` needs to point to a file, which does not exist (e.g. `master_wavelength_manual.fits`). Then the python scrip can be run normally. It will stop after searching for the lines in the emission line spectra. The order and pixel position of all identified lines are stored in the file given in parameter `logging_found_arc_lines`. This file can be opened and the corresponding wavelengths can be added to some of the lines. I have found that providing 1.5 lines per order is enough for the pipeline to do the rest. The lines should cover as much of the CCD image as possible.

The correlated data needs to be saved into file `arc_lines_wavelength.txt` with the following tab-separated entries:

- Aperture (starting at 0)
- Real order (bigger than 0, usually between 40 and 120)
- Pixel
- Wavelength
- (Type of the line (e.g. ThI, NeII) ).

Afterwards the script can run again and will use this data to create a new wavelength solution.

## 5.2 Setting up a new account or a new camera

When setting up a new camera or a new account paths and parameters need to be adjusted. The following list explains necessary changes and how to determine parameters which need to be changed.

**badpx\_mask\_filename** : Insert path to the bad-pixel mask. Use 'NA', if no bad-pixel mask is needed.

**reference\_catalog** : Change the path to the catalogue of the reference lines. The file contains one entry per line, each entry consists of tab-separated wavelength, line strength, and element. Line strength can be empty.

**rotate\_frame, flip\_frame** : The CCD image needs to be aligned in a way that the cross-dispersion axis is along vertical direction and the dispersion axis is along the horizontal direction, AND that the wavelength in dispersion direction is increasing with higher pixel number by providing the necessary angle for `rotate_frame`. If necessary the image can be flipped (`flip_frame = True`) after the rotation so that the wavelength in cross-dispersion direction decreases with increasing pixel number (blue orders on the right and the red orders on the left).

**width\_percentile** Only pixel with more flux than the value given in `width_percentile` (of the maximum flux) will be extracted. The extraction width can be varied later using the parameters `extraction_width_multiplier` and `arcextraction_width_multiplier`. The covered area can be checked in the logged images given in the parameters `logging_traces_im` and `logging_arctraces_im`.

**bin\_search\_apertures** : In order to speed up the search for the traces, heavy binning should be applied. However, the value for the cross-dispersion axis should be chosen in a way, that neighbouring orders are still separated. This can be checked in the images `flat_binned.fits`.

**arcshift\_side** : Position of the calibration traces compared to the science traces. Can be 'left', 'right', or 'center'. The latter option implies, that only one fiber exists.

**raw\_data\_imtyp\_keyword, raw\_data\_imtyp\_\*** : Put the correct information for the image type header key here.

**raw\_data\_exptime\_keyword, raw\_data\_dateobs\_keyword** : Use the right header keywords for the exposure time and the observation date. The format should be `%Y-%m-%dT%H:%M:%S.%f` or<sup>1</sup> `%Y-%m-%dT%H:%M:%S`. More formats can be added in the procedure `get_obsdate`.

**raw\_data\_timezone\_cor** In the unlikely case, that the date and time stored in `raw_date_timezone_cor` is not UTC, the time zone correction can be given here. Numbers will be positive east of UTC and negative west of UTC, e.g. +7 for Bangkok time or -10 for Hawaii.

**standard\_calibs\_create** : Define the standard CCD processing, if necessary (see Section 4.4 for options).

---

<sup>1</sup><https://docs.python.org/2/library/datetime.html#strftime-and-strptime-behavior>

**site, altitude, latitude, longitude** : If the site information is not stored in the header (see Section 4.6), then the information can be stored in the configuration file.

Further settings are described in Section 4.3.

### 5.3 Create a bad pixel mask

The bad pixel masked used by the pipeline is fits file which consists of a 2 dimensional image consisting of '1' for good data and '0' for bad pixels. It can be created by the script `create_badpx_mask.py`, but this is script is really work in progress at the moment.

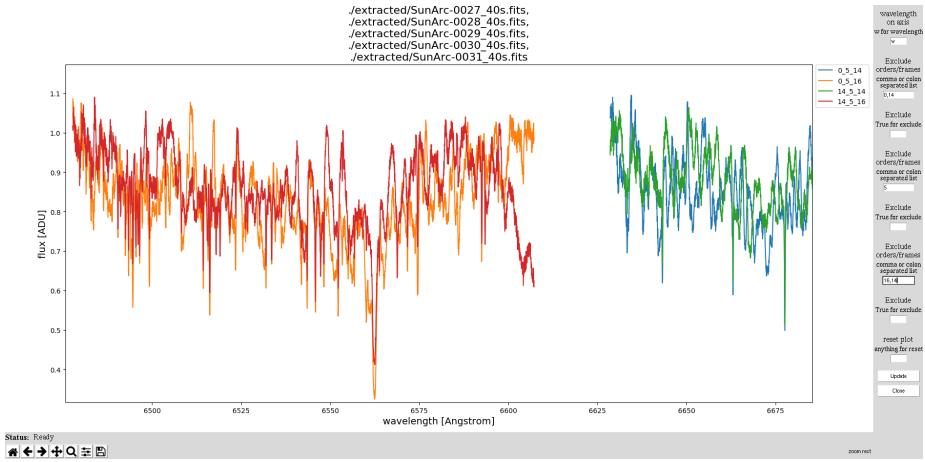


Figure 6: An example of how to plot the extracted data. Shown are the solar spectra of entry 0 and 14 in the plot\_files.lst file. The graph shows the continuum normalised flux (data type 5), plotted over the wavelength (w). Only apertures 14 and 16 are shown.

## 6 Post-extraction analysis

Once the science spectra were extracted these files can be analysed more in detail.

### 6.1 Plotting the data

To plot the results the python program `plot_img_spec.py` can be used. This script read the files given in the file `plot_files.lst` and will plot at the beginning all data. Further limitations of the plotted data can be done using the UI. For a plot in different axis the first text box can be used. See the list below for options. For plotting only data from a subsection of files, the first text boxes 'Which file' and 'Exclude' need to be used. For plotting only a subsection of data types the second text boxes with 'Which data' and 'Exclude' are used to define this. For plotting only some apertures, the last text boxes are used (see an example in Fig. 6 and 7).

The following options for axis plotting exist:

- (empty): The flux is plotted against the pixel along the CCD in dispersion direction.
- w** : The flux is plotted against the wavelength (stored in data type 0).
- f** : The Fourier transformation is plotted against the period (in pixel).
- l** : The Lomb-Scargle-Periodogram is plotted against the period (in pixel).
- wl** : The Lomb-Scargle-Periodogram is plotted against the period (in wavelength). If you look for periodic signals in the spectrum, this is what you very probably should use.

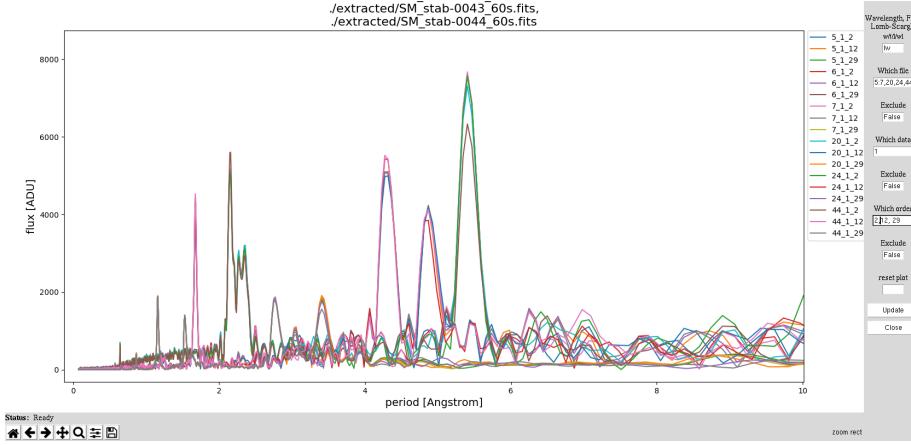


Figure 7: Another example of how to plot the extracted data. Shown is the Lomb-Scargle Periodogram for the white light spectra of entry 5 to 7, 20, 24, and 44 in the plot.files.lst file. The extracted spectra (data type 1) of apertures 2, 12, and 29 and their corresponding wavelength solution was used as basis for the calculations.

## 7 Extracting data from other spectrographs

### 7.1 Extracting HARPS data

#### 7.1.1 Getting the data

Search and download the data from [http://archive.eso.org/eso/eso\\_archive\\_main.html](http://archive.eso.org/eso/eso_archive_main.html). Check only 'HARPS/LaSilla'. For example data select night '2015 07 30' to find data of Ceres or use 2010 10 23/25 for HD 10180 observation. Then download the data (save the download request script, cd to the folder, and run `sh downloadRequest<Number>script.sh`). The raw data (\*.fits.Z) can be extracted with:

```
gunzip *
cat <<EOT > extract_single_chip.py
#!/usr/bin/python
# -*- coding: utf-8 -*-
import os
from astropy.io import fits

os.system('mkdir -p red_chip blue_chip')
for entry in os.popen('ls -1 *.fits').readlines():
    im_r = fits.getdata(entry[:-1],2)
    im_b = fits.getdata(entry[:-1],1)
    im_head_r = fits.getheader(entry[:-1],2)
    im_head_b = fits.getheader(entry[:-1],1)
    im_head0 = fits.getheader(entry[:-1],0)
    fits.writeto('red_chip/'+entry[:-1], im_r, \
                im_head0, overwrite=True)
    fits.writeto('blue_chip/'+entry[:-1], im_b, \

```

```

        im_head0, overwrite=True)
EOT
python extract_single_chip.py
```

### 7.1.2 Preparing for data extraction

The following replacements are necessary in the conf.txt file:

- subframe = [4096,2048,0,50]
- rotate\_frame = 180
- bin\_search\_apertures = 20,2
- arcshift\_side = left
- original\_master\_arc\_solution\_filename = master\_arc\_solution\_manual.fits.  
The file arc\_lines\_wavelength.txt needs to be created (Section 5.1).
- arc\_lines\_catalog = <path>reference\_lines\_ThAr-HARPS\_Ceres.txt
- use\_catalog\_lines = ThI, ArI, ?, NoID, ArII, ThII
- raw\_data\_imtyp\_keyword = OBJECT
- raw\_data\_imtyp\_bias = BIAS,BIAS
- raw\_data\_imtyp\_sflat = LAMP,DARK,TUN
- raw\_data\_imtyp\_flatarc = LAMP,LAMP,TUN
- raw\_data\_imtyp\_arc = WAVE,WAVE,THAR2
- bias\_calibs\_create\_g = subframe
- trace1\_calibs\_create\_g = subframe, bias
- trace2\_calibs\_create\_g = subframe, bias
- arc\_calibs\_create\_g = subframe, bias
- flatarc\_calibs\_create\_g = subframe, bias
- extract\_calibs\_create\_g = subframe, bias
- polynom\_order\_traces = [2,3]
- polynom\_order\_intertraces = 2,3

Afterwards, `prepare_file_list.py` and `reduction_day.py` can be run as described above.

### 7.1.3 HARPS data reduced by the CERES pipeline

To compare the results from the EXOhSPEC pipeline, it can also be compared with the results from the CERES pipeline (<https://ui.adsabs.harvard.edu/#abs/2017PASP..129c4002B/abstract>, <https://github.com/rabrahm/ceres>). The extraction can be run with:

```

cd ~/software/ceres-master/harps/
python harpspipe.py /data/ronny/Reduced/20180313_harps_sol_ceres_red/ -do_class
```

The pipeline produces the following files:

```

spec.co.B.fits.S, spec.co.R.fits.S
spec.ob.B.fits.S, spec.ob.R.fits.S
```

**proc/results.txt** Results of the cross-correlation, including the radial velocity.

**proc/\*.fits** The reduced images, in the similar format as produced by the EXOhSPEC pipeline.

Please note that the CERES pipeline uses both the red and the blue HARPS chip in order to determine the radial velocities (the data of the blue chip can be avoided by changing `spec` into `spec[:, :26, :]` in `harpspipe.py`, lines 1507 and 1530). Furthermore, a cosmic ray correction is applied. The extracted spectra have the same features as when extracting data with the EXOhSPEC pipeline. A wavelength shift (depending on the order) is visible between the spectra. This translates into a RV shift of  $\Delta RV \approx 79$  km/s (independent of order, no instrumental velocity drift were applied to EXOhSPEC).

#### 7.1.4 HARPS data reduced by the ESO Phase 3 pipeline

The reduced data can be found at: [http://archive.eso.org/wdb/wdb/adp/phase3\\_spectral/form](http://archive.eso.org/wdb/wdb/adp/phase3_spectral/form). Select 'ESO-3.6' and 'HARPS'. For example data use Date Obs '2015-07-30..2015-08-01' or the Run/Program ID from the results of the query for the raw data (make sure to use the ID of the science, not the calibration data). The download the data. The reduced data (\*.tar) can be extracted using

```
cat *.tar | tar -xvf - -i
```

The data is reduced in the following way<sup>2</sup>:

- bias subtraction via overscan region
- dark subtraction via values stored in a database
- order extraction into 'e2ds' files
- flat fielding (flat/blaze function is used as flat)
- wavelength calibration
- order matching into 1D files 's1d'
- cross correlation for RV - $\zeta$  'ccf' and 'bis' files
- correction for instrumental drift using the calibration fiber

The following files are available:

**ADP.\*.fits (primary file)** Table with wavelength, flux (, and error) in 0.01 Å steps. The files are in random order, check DATE-OBS to sort them.

**HARPS.\*\_bis\_<sptype>\_A.fits** Bisector from cross correlation with the <sptype> mask.

**HARPS.\*\_ccf\_<sptype>\_A.fits** Cross correlation function matrix for mask for <sptype>.

**HARPS.\*\_ccf\_<sptype>\_A.tbl** Cross correlation function summary table (ASCII) with extracted radial velocity per each order.

---

<sup>2</sup><http://www.eso.org/rm/api/v1/public/releaseDescriptions/72>

**HARPS.\*\_e2ds\_<fibre>.fits** 2D extracted spectrum with wavelength solution in the header.

**HARPS.\*\_INT\_GUIDE.fits** Image from the integrated guiding camera (star centred on the fiber).

**HARPS.\*\_s1d\_<fibre>.fits** 1D extracted full spectrum, wavelength calibrated, in the solar system barycentric frame.

The wavelength solution in the `e2ds` files is stored in the header<sup>3</sup>. The order numbering starts at 0 (bluest order) and the pixel numbering **probably** starts at 0, too (bluest pixel). The `e2ds` files can be converted to the EXOhSPEC format by using the script `reduced_harps_to_exohspec.py`.

**Results** The extracted calibration spectrum and the science spectrum are very similar. Emission and absorption lines are at the same wavelength. The extracted flux is about 10-12% lower and the noise slightly higher.

Few problems occur when comparing the data of reduced with this pipeline and the data from the ESO Phase 3 pipeline:

1. The bias correction can't be performed, as usually only one bias frame is available.
2. Overscan correction is not implemented in the pipeline.
3. No cosmic ray removal is applied.
4. Different flat fielding.
5. Traces have a fixed width, therefore the S/N of the extracted data might decrease slightly.
6. Only one CCD is processed at a time, which might decrease the precision of the wavelength solution.
7. The wavelength shift between science and calibration fiber is not determined (yet).

### 7.1.5 Analysing the reduced HARPS data with TERRA

For more information about TERRA please refer to the literature<sup>4</sup>. The following files are used by TERRA:

**HARPS.\*\_bis\_<sptype>\_A.fits** Bisector from cross correlation with the <sptype> mask.

**HARPS.\*\_ccf\_<sptype>\_A.fits** Cross correlation function matrix for mask for <sptype>.

**HARPS.\*\_e2ds\_<fibre>.fits** 2D extracted spectrum with wavelength solution in the header.

*Todo: more*

---

<sup>3</sup>See page 22 in <https://www.eso.org/sci/facilities/lasilla/instruments/harps/doc/DRS.pdf>

<sup>4</sup><https://ui.adsabs.harvard.edu/#abs/2012ApJS..200...15A/abstract>

## 7.2 Extracting MRES data

The testing of the pipeline was done for the MRES data taken on 2018-11-27.  
The following replacements/insertions are necessary in the conf.txt file:

- subframe = [12091,511,380,0]
- rotate\_frame = 270
- arcextraction\_width\_multiplier = 1
- maxshift = 5
- bin\_search\_apertures = 10,1
- bin\_adjust\_apertures = 2,1
- width\_percentile = 10
- arcshift\_side = center
- raw\_data\_imtyp\_bias = NA
- raw\_data\_imtyp\_dark = NA
- raw\_data\_imtyp\_flat = NA
- raw\_data\_exptim\_keyword = EXPOSURE
- raw\_data\_dateobs\_keyword = DATE
- arc\_calibs\_create\_g = subframe, <...>
- trace1\_calibs\_create\_g = subframe, <...>
- trace2\_calibs\_create\_g = subframe, <...>
- extract\_calibs\_create\_g = subframe, <...>
- flatarc\_calibs\_create\_g = subframe, <...>
- wavelengthcal\_calibs\_create\_g = subframe, <...>
- extracted\_bitpix = -32

## A Module dependencies if not using Anaconda

This program relies on the following modules in python 2.7

- numpy
- os
- sys
- time
- datetime
- operator
- copy
- random
- warnings
- tqdm
- pickle
- json
- astropy
- scipy
- matplotlib
- Tkinter
- collections
- ephem
- math.radians
- statsmodels.api

## A Short introduction to reduction of CCD images

The data stored in CCD images is affected by the physical parameters of the individual pixels and way the readout electronics work. Therefore each scientific frame needs to be corrected (pixel by pixel) with the formula:

$$\text{reduced Science} = \frac{\text{raw Science} - \text{master Bias} - \text{master Dark}}{\text{master Flat}}. \quad (1)$$

The master Dark should have the same exposure time as the Science frame and the master Flat should be normalised in order to keep the flux levels. To create the master Dark and master Flat, the following steps are necessary:

$$\text{master Dark} = \text{raw Dark} - \text{master Bias} \quad (2)$$

$$\text{master Flat} = \text{raw Flat} - \text{master Bias} - \text{master Dark}. \quad (3)$$

The master Dark used for the creation of the master Flat should have the same exposure time as the Flat and needs to be corrected with the Bias. The master Bias in each of the formulas can be different.

In order to decrease the noise levels, each of the master file should be created by median combining several (corrected) images together. If the brightness of the individual flats vary, then the flats should be weighted by their median flux before combining them. Combining all steps leads to the formula

$$\text{reduced Science} = \frac{\text{raw Science} - B_S - (D_S - B_{D_S})}{F - B_F - (D_F - B_{D_F})}, \quad (4)$$

where  $B_S$  is the master Bias, created by combining biases taken at a similar time as the science frame,  $D_S$  is the master Dark, created of darks with the same exposure time as the science frame. In case the bias level and noise don't change during the night, then Equation 4 changes to

$$\text{reduced Science} = \frac{\text{raw Science} - D_S}{F - D_F}. \quad (5)$$

In case the dark level is negligible and the Bias level is constant, then Equation 4 changes to

$$\text{reduced Science} = \frac{\text{raw Science} - B}{F - B}. \quad (6)$$

In the pipeline always the individual frames are corrected before frames are combined.