User Manual for Data Reduction of the EXOhSPEC Instrument Working version: V0.2.0

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

February, 2018

Contents

1 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, and perform further calibration step.

2 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).

2.1 With Anaconda and python 2.7

Anaconda can be downloaded from https://www.continuum.io/downloads. 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).

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`).

2.1.1 Additional modules

Once Anaconda is installed one other module is needed (tqdm). If Anaconda is installed correctly then pip can be used to install this module

```
pip install tqdm
```

2.2 Necessary python 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/v0.1.0.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 This needs to be run in order to find an initial wavelength solution (will be included in reduction_day.py later).

create_badpx_mask.py Script to create a bad pixel mask.

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

2.3 General information about the way the pipeline works

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`, 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.

Finding further information

- The pipeline logs the execution of procedures and necessary information in a logfile (standard: `logfile`). 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 4.
- 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`.

3 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:

$$\text{reduced Science} = \frac{\text{raw Science} - \text{master Bias} - \text{master Dark}}{\text{master Flat}}\,. \tag{1}$$

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:

$$\text{master Dark} = \text{raw Dark} - \text{master Bias} \tag{2}$$

$$\text{master Flat} = \text{raw Flat} - \text{master Bias} - \text{master Dark}\,. \tag{3}$$

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

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

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

$$\text{reduced Science} = \frac{\text{raw Science} - D_S}{F - D_F}\,. \tag{5}$$

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

$$\text{reduced Science} = \frac{\text{raw Science} - B}{F - B}\,. \tag{6}$$

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

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:

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

   arc 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 60 s ThAr).

   arc_s : Short exposure in order to find the center of the saturated lines in arc_l (standard: 5x 10 s ThAr taken between the arc_l images).

   flatarc : File which contains the flat in the science fiber and the arc in calibration fiber (standard: 5x files with the white light flats in science fiber and ThAr in the calibration 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.

3. Creating a map of the leaking light between the science orders (background map).

4. Finding the traces of the calibration orders (→ Fig. 2).

5. Create the wavelength solution for the night (→ Fig. 3 and 4). This is based on the solution of the previous data set, which is adjusted according to the given parameters.

6. Extract the flat and normalise it.

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 orders for the science fiber

   d) Extraction of the orders for the calibration fiber

   e) Finding the shift between the emission lines in the calibration spectra and the lines used for the wavelength solution, and 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 some general measurements which are stored in the header.

i) Write the file with all extracted data into the folder given in the parameter `path_extraction` (standard: `extracted`).

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 safe 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:

- *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 **flat** for automatic processing.
- Flat (3x, white light source through the science fiber), with the filename containing **sflat**. 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. 60 s) exposure time). The filename should start with **arc** and the science fiber should be dark for this data.
- FlatArc (11x, white light source and calibration lamp), the filename should be **flatarc**.
- Bias (11x) and/or Darks (11x, for the exposure time of the true Flat and FlatArc)

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.

   raw_data_path : Folder to the raw data.

   original_master_order_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). 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 .`

After checking the results (see Section 4.7), one might want to remove wrongly identified orders. This can be done in a graphical user interface by running `remove_orders.py`. The old files will be moved into a sub-folder and the necessary 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`.

flat : Apply a master flat. The necessary parameters are created automatically by the script `prepare_file_list.py`.

background : Applies background correction using the filename given in the parameter `background_filename`.The background image will be scaled by the exposure time of the scientific image, before its subtracted from the scientific image.

localbackground : Applies background correction by fitting a 2d polynomalial against the current reduced image, excluding the area of the science and calibration traces. The fit is then removed from the image.

The following parameters change the way, several CCD images are combined:

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

combine_sum : Images are summed up instead of median-combining them.

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_exptim_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).

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

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 `bias`	bias	bias
filename contains `dark`	dark	dark
filename contains `flat` but not `sflat`	flat	flat
`raw_data_imtyp_keyword` equals `raw_data_imtyp_flat` and filename dosn't contain `sflat`	flat	flat
filename contains `arc`	–	wave
`raw_data_imtyp_keyword` equals `raw_data_imtyp_bias`	bias	bias
`raw_data_imtyp_keyword` equals `raw_data_imtyp_dark`	dark	dark
none of the above and the filname doesn't start with `arc`	science	–

- Type of light for the science fiber
- Type of light for the calibration fiber
- Exposure time in seconds
- Observation time
- 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 induvidually, 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 Format of the extracted files

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 orders. The data in the file is stored in a 3D array in the form: data type, order, 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 order and pixel.
1. Extracted spectrum without any modification.

2. (Measurement of the error the extracted spectrum)

3. Flat corrected spectrum, calculated by dividing the extracted spectrum and the normalised flat spectrum.

4. (Error the flat 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.

4.7 Results from the pipeline

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_order_filename` (standard: `master_orders.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 order.
- Parameters of a polynomial fit to the trace (given in parameter `polynom_order_orders`), e.g. 5 values if the trace is fitted with a polynom of order 5.
- 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: the width from the center to the left, the width from the center to the right, and the Gaussian width.

The identification of the traces can be checked easily in the file given in parameter `logging_orders` (standard: `orders_in_master_flat.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 background map, 2d image is stored in the file given in parameter `background_image_filename` (standard: `background.fits`). Additionally a mask which provides the pixel which are used to create the background map

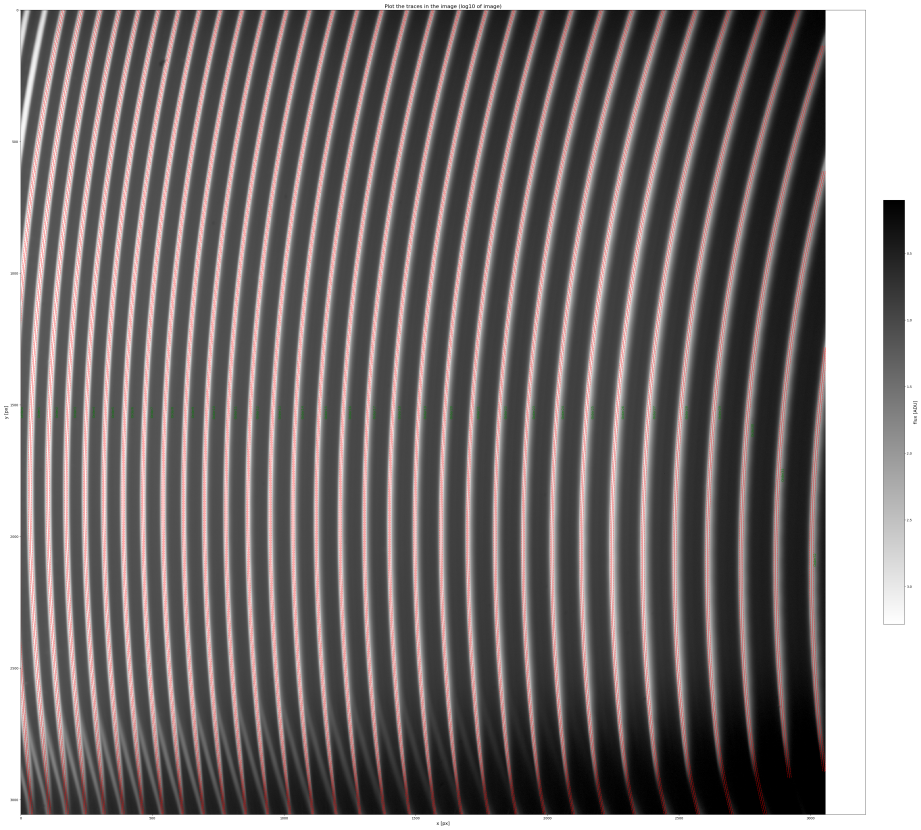

Figure 1: Reduced CCD image of a white light flat (log10 gray scale) with the marked traces of the identified scientific orders (red). The extraction width (determined from the Gaussian width of the order multiplied with the value in parameter **extraction_width_multiplier**) is given in dashed lines.

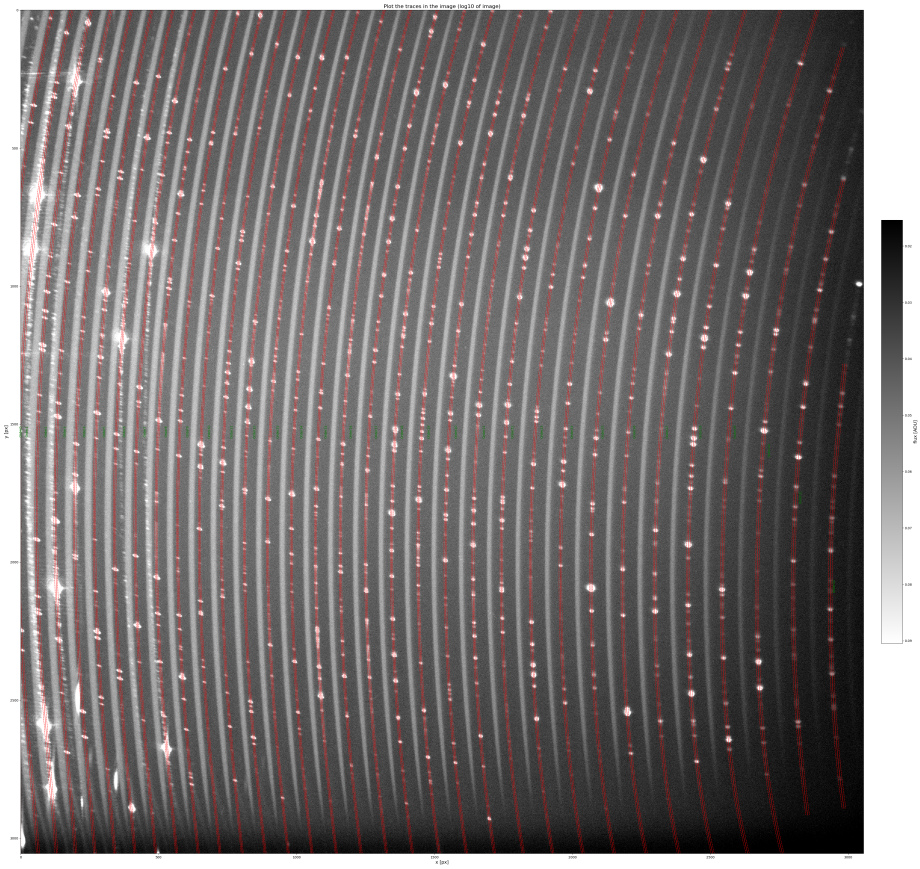

Figure 2: Reduced CCD image of a ThAr exposure (log10 gray scale) with the marked traces of the identified calibration orders (red). The extraction width is given in dashed lines. The extraction width (determined from the Gaussian width of the order multiplied with the value in parameter **arcextraction_width_multiplier**)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

is stored in the file given in parameter `background_px_filename` (standard: `background_px.fits`).

Step 4: The traces of the orders of the calibration fiber are stored in the file given in parameter `master_orderarc_filename` (standard: `master_ordersarc.fits`). 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_arcorders` (standard: `arcorders_in_master_arc.png`). An example is show in Fig. 2.

Step 5: The wavelength solution is stored in the file given in parameter `master_arc_solution_filename` (standard: `master_arc_solution.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.
- Parameters of a polynomial fit to the trace, e.g. 4 values if the dispersion axis is fitted with a polynom 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 524 lines. The standard deviation of the fit is 0.0054 Angstrom. A 2d
polynom fit with 4 orders along the traces and 4 orders perpendicular to the orders
was used. With this solution, the offset to real orders is 72. With this offset, the
standard deviation of the residuals between the central wavelengths and the grating
equation is 0.1295 Angstrom. Using the original solution gives an offset of 72.
ord cenwav minwav maxwav ranwav Ang/px name numb gausswi gausswi min_   max_   range_reflin
                                              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:

order Order of the trace, starting with 0 for the reddest order ($\tilde{m}$). 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 $y = (m_0 + \tilde{m})\lambda_c$, were λ_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 $\frac{\mathring{A}}{\text{px}}$ 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 Minumum 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)

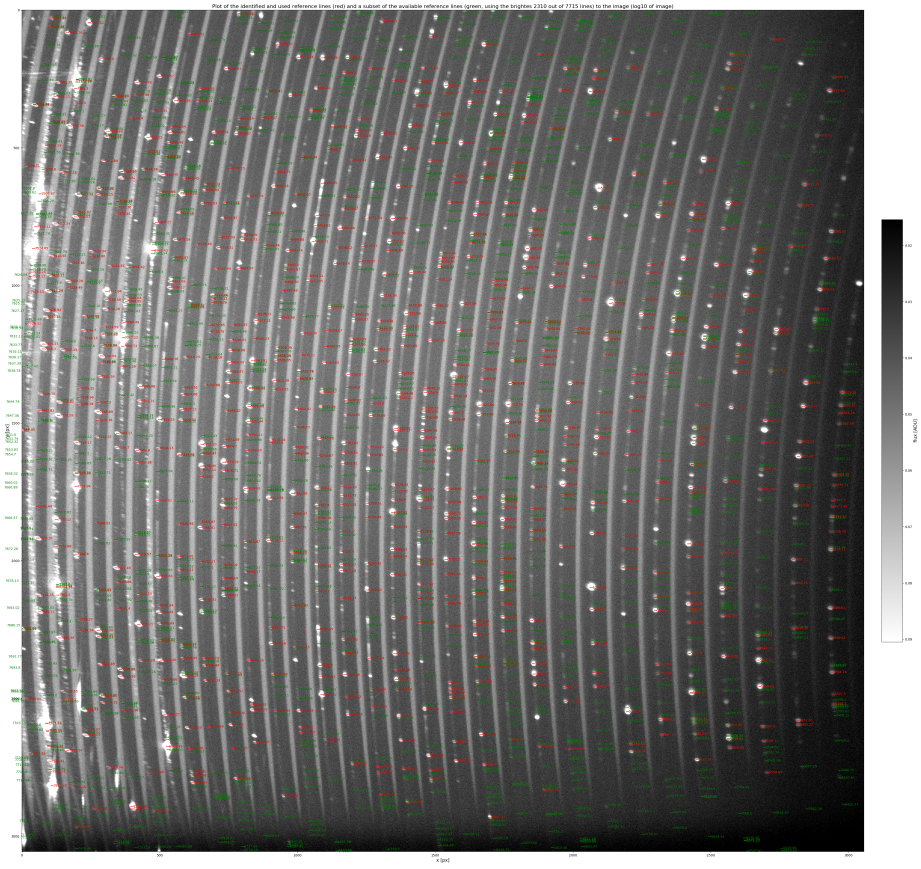

Figure 3: Reduced CCD image of a ThAr exposure (log10 gray scale) with the identified lines from the reference catalogue (red). The remaining lines of the reference catalogue, which weren't used for fitting the solution are shown in green. Only this set of data was used to create the wavelength solution.

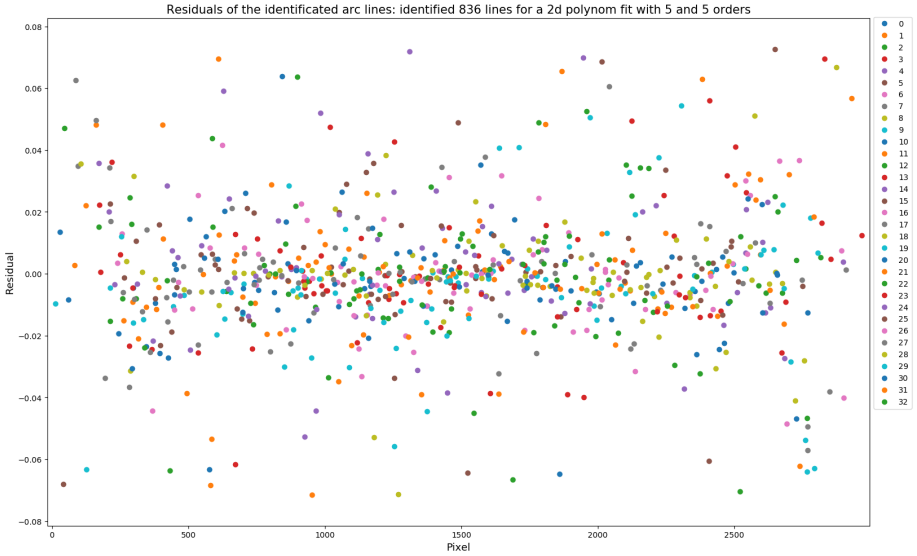

Figure 4: Residuals between the identified cataloge lines and the wavelength solution. Colorcoded are the different orders.

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

Step 6: 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 4.6

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:

- Order (starting at 0)
- Real order (bigger than 0, usually between 60 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).

extraction_width_multiplier, arcextraction_width_multiplier : The spectrum will be extracted by summing up the data between the center of a trace and `extraction_width_multiplier` times the Gaussian width of the order to either side of center. The covered area can be checked in the logged images given in the parameters `logging_orders` and `logging_arcorders`. In case of a real Gaussian profile, the full width at half maximum and Gaussian width are connected with $FWHM = 2.35482 \cdot w_{\mathrm{Gauss}}$.

bin_search_orders : In order to speed up the search for the traces, heavy binning should be applied. However, the value for the cross-dispersion axis should to 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 later 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_exptim_keyword, raw_data_dateobs_keyword : Use the right header keywords for the exposure time and the observation date.

standard_calibs_create : Define the standard CCD processing, if necessary (see Section 4.4 for options).

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.

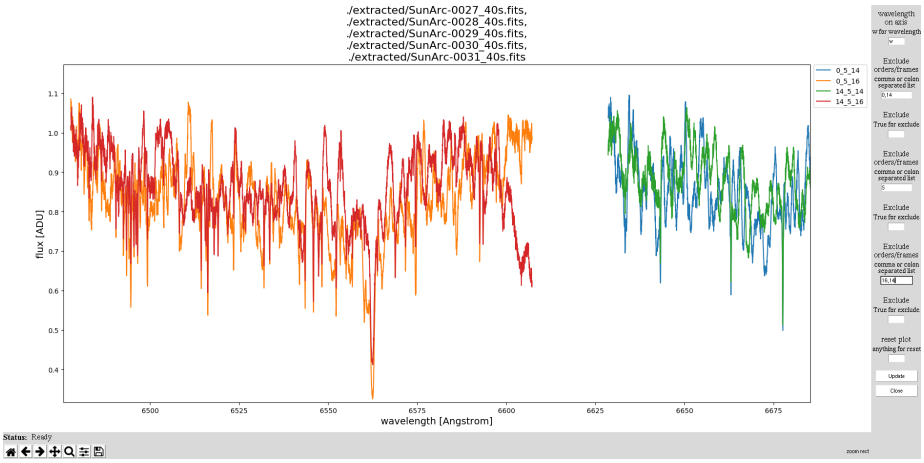

Figure 5: An example of how to plot the extracted data. Shown are the solar spectra of entry 0 and 14 in the plot_files.lst file. The graph shows the continuum normalised flux (data type 5), plotted over the wavelength (w). Only orders 14 and 16 are shown.

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 orders, the last text boxes are used (see an example in Fig. 5 and 6).

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.

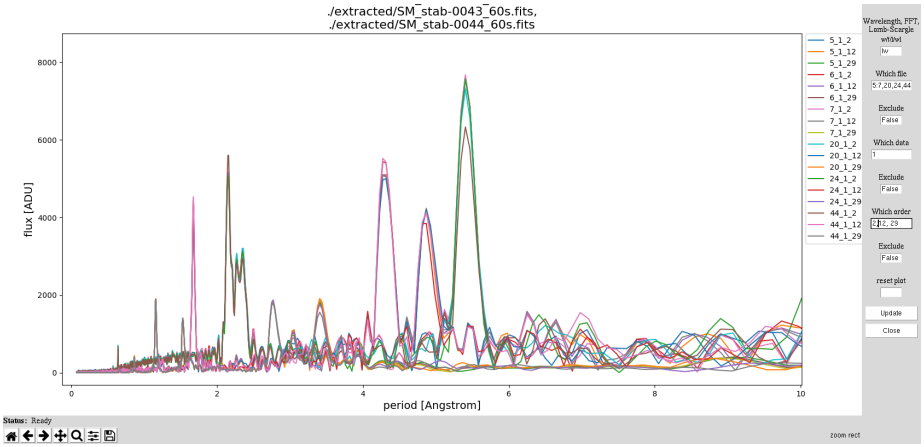

Figure 6: Another example of how to plot the extracted data. Shown is the Lomb-Scargle Periodogram for the white light spectra of entry 5 to 7, 20, 24, and 44 in the plot_files.lst file. The extracted spectra (data type 1) of order 2, 12, and 29 and their corresponding wavelength solution was used as basis for the calculations.

6.2 Finding the radial velocity

The radial velocity measurement is based on the *CERES* pipeline. Thereby a template is cross correlated with the scientific spectrum. For the analysis the following steps need to be run:

```
ls extracted/*.fits > analysis_files.lst
python <path to script>/find_rv.py | tee RV_logfile
```

7 Extracting HARPS data

7.1 Getting the data

Search and download the data from http://archive.eso.org/eso/eso_archive_main.html. For example data select night '2015 07 30' and check only 'HARPS/LaSilla'.

The reduced data can be found at: http://archive.eso.org/wdb/wdb/eso/repro/form. For example data select night '30 07 2015'.

The raw data (*fits.Z) can be extracted with

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

os.system('mkdir -p red_chip')
for entry in os.popen('ls -1 *.fits').readlines():
    im = fits.getdata(entry[:-1],2)
    im_head = fits.getheader(entry[:-1],2)
    im_head0 = fits.getheader(entry[:-1],0)
    fits.writeto('red_chip/'+entry[:-1], im, im_head0, overwrite=True)
EOT
python extract_red_chip.py
```

The reduced data (*.tar) can be extracted using

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

7.2 Preparing for data extraction

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

- `subframe = [4096,2048,0,50]`
- `rotate_frame = 180`
- `bin_search_orders = 20,2`
- `arcshift_side = left`
- `raw_data_imtyp_keyword = OBJECT`
- `raw_data_imtyp_bias = BIAS,BIAS`
- `raw_data_imtyp_sflat = LAMP,DARK,TUN`
- `raw_data_imtyp_arc = WAVE,WAVE,THAR2`
- `bias_calibs_create_g = subframe`
- `sflat_calibs_create_g = subframe, bias`
- `arc_calibs_create_g = subframe, bias`
- `flatarc_calibs_create_g = subframe, bias`
- `extract_calibs_create_g = subframe, bias`
- `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 = NeI, UI, ThI, ArI, ?, NoID, ArII, ThII`

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

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

if python2:

- Tkinter
- collections

if python3:

- tkinter