User Manual for Data Reduction of the EXOhSPEC Instrument
Working version: v0.4.1

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

April, 2019

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, measure the radial velocity, and perform further calibration steps.

If you use the barycentric corrected radial velocities, the barycentric velocity, or the BJDTDB, please cite Kanodia and Wright 2018[1].

2 For Beginners in Echelle Spectroscopy or Programming

This Pipeline was intended to be written as a black box for the unskilled user, however a basic understanding of echelle spectroscopy and data reduction of digital astronomical images is necessary in order to use it.

2.1 Data reduction of CCD or CMOS images (photometry or spectroscopy)

Please refer to Howell (1976,2006)[2] or check Chapter C.

2.2 Echelle spectroscopy

Echelle spectrographs image the disperse the light into a 2-dimensional plane by using a high-dispersing grating and a low-dispersing prism (or grating). The low-dispersing element is thereby necessary to separate the individual orders produced by the high-dispersing grating. Some of the wavelengths can be covered by several orders. An example for an echelle spectrum of a black body is shown in Fig. 3 with the individual orders going up to down and order number increasing from left to right.

In order to work with the spectra, the following steps have to be done:

- Trace the orders.
- Correlate wavelengths to pixel.
- Extract the spectra.
- Apply instrument specific corrections.

2.3 Programming (Python)

- Lines starting with one or more # are comments, which will be ignored by the pipeline.
- Any text is case-sensitive (except when mentioned otherwise).
- Empty lines and the amount of spaces normally doesn't matter.

[1] https://iopscience.iop.org/article/10.3847/2515-5172/aaa4b7

[2] https://www.cambridge.org/core/books/handbook-of-ccd-astronomy/97D3D910788D44D11394B3B57C3FA743

3 Changelog (recently)

Please found the changes of older versions in Chapter A.

v0.4.0

- Included the comparison between the wavelength solutions in science and calibration fiber.
- Improved radial velocity analysis using Terra.
- Renamed `arc_l` and `arc_s` into `cal2_l` and `cal2_s`.
- Better way of getting the object name from the filename.
- Added parameter `path_ceres` to conf.txt.
- Added parameter `terra_jar_file` to conf.txt.
- Renamed parameter `path_rv` to `path_rv_ceres`.
- Renamed parameter `csv_path` to `path_csv_terra`.
- General bug fixing.

v0.4.1

- Switched to barycentric correction (radial velocity and BJDTDB) from https://github.com/shbhuk/barycorrpy.
- Added parameter `raw_data_mid_exposure_keys` to conf.txt (optional at the moment).
- Added parameter `path_harpsformat` to conf.txt
- Changed parameter `raw_data_path` to `raw_data_paths`, allowing list of folders.
- Renamed parameter `maxshift` to `maxshift_orders`.
- Renamed parameter `update_widths` to `update_width_orders`.
- Renamed parameter `master_flat_spec_norm_filename` to `master_blaze_spec_norm_filename`.
- Renamed parameter `raw_data_imtyp_flatarc` to `raw_data_imtyp_blaze`.
- Script `prepare_file_list.py` changed to allow for easy GUI development. Files can be assigned directly to calibration within the script.
- Time format in the file from parameter `configfile_fitsfiles` created by `prepare_file_list.py` changed.
- Improved determination of shift between current frame and the reference frame.
- Improved Error messages.
- Improvement of the way how the new header values are stored.
- Improvement of the GUI and plotting the spectra.

4 Quick starting Guide

4.1 Setting up a new instrument

The pipeline reads the parameters from the configuration file `conf.txt`. This section explains in detail how to reduce the data for a new spectrograph.

1. General parameters:
   - Create a new folder and copy the `conf.txt` into that folder and open the file in an editor.
   - Change parameter `raw_data_path` to point to the folder or folders with your raw or pre-reduced fits-files. The data can be stored in sub-folders.
   - Set the parameter `badpx_mask_filename` if a bad-pixel-mask exists (see Chapter 4.7). Otherwise leave empty or 'NA'.
   - Modify the header-parameters, if necessary (see Cahpter 4.6).
   - Adjust `rotate_frame` and `flip_frame` so that the orders are up to down (blue wavelength on top) and red orders are on the right. The pipeline will first rotate and then flip, the flip is swapping left and right.
   - Adjust parameter `subframe` to the full CCD size or a subframe, if only part of the CCD should be used.
2. Finding the traces of the science fiber:
   - Set parameter `original_master_traces_filename =` (empty), or to a non-existing file.
   - To increase signal to noise and to speed up the search for the traces of the orders, binning as given in the parameters `bin_search_apertures` (rough estimate) and `bin_adjust_apertures` (fine tuning) can be adjusted. Please note that after binning, the orders should still be well distinguished from each other.
3. Finding the traces of the calibration fiber:
   - The side of the calibration traces compared to the science traces is given in parameter `arcshift_side`. Set it to center for a single fiber spectrograph.
4. Create a new wavelength solution (in case no wavelength solution is required, set `master_wavelensolution_filename` to `pseudo` and skip the other steps under this bullet point):
   - Change the path to the catalogue of the reference lines (`reference_catalog`), if necessary. The file must contain one entry per line, each entry consists of tab-separated wavelength, line strength, and element. Line strength can be empty.
   - Set parameter `original_master_wavelensolution_filename = master_wavelength_manual.fits`.
   - To find a wavelength solution a file which provides the wavelength for some (extracted) pixel and wavelengths needs to be given (called *pixel_to_wavelength.txt*. Until a GUI is available within the pipeline, a way to create the list is described in Chapter 4.4.

- The number of degrees of freedom for the wavelength solution (2-dimensional polynomial fit) can be adjusted with `polynom_order_traces` (polynomial orders along the dispersion direction) and `polynom_order_intertraces` (polynomial orders between the traces).

5. Setting up the header keywords:
   - Adjust the parameters starting with `raw_data_*`. The example script shows the settings for data taken with MaximDL and for data from the HARPS spectrograph.
6. Setting up the data reduction steps:
   - Adjust the parameters `*_calibs_create_g` to define the reduction steps which should be applied. Please refer to Chapter 4.3 for further information. Some of the `*_calibs_create_g` parameters might not be relevant, e.g. if no *dark* or *rflat* (real flatfield) corrections should be applied. An example for the parameters with additional explanation is shown in Figure 1.
7. Setting up the observatory site:
   - The parameters `site`, `altitude`, `latitude` (negative for west), and `longitude` need to be set, if the information is not stored in the fits-header (see also Chapter 5.4).
8. Create the file *object_list.txt*: See Chapter 4.5.
9. Run the scrip `prepare_file_list.py` (see Chapter 4.6 below for more information):
   - Define what files should be what calibration.
   - Please note that you can use already reduced images. In this case the reduction steps above should be empty for the file type (e.g. dark or real flats) to avoid a second application of the correction.
10. Run the scrip `reduction_day.py`:
    - Afterwards: check the output in the `logfile`, the images in the logging path, or in results in the extracted files.

4.2 Reducing data with the same setup for different nights

After all the calibration steps have been performed for a given instrument (see Chapter 4.1), this information can be used to extract data of different nights more easily, as long as the instrument itself stays untouched. Here it is assumed that files to find the traces of science and calibration fiber, files to adjust the wavelength solution, and files to find the blaze function are available.

- Copy the *conf.txt* from the previous night as most modifications will be made there already.
- Set parameter `original_master_traces_filename` to the file *master_traces_sci.fits* in the previous night (contains the properties of the traces).

```
bias_calibs_create_g            = []
dark_calibs_create_g            = [subframe_nooverscan, badpx_mask]
rflat_calibs_create_g           = [subframe, badpx_mask, dark, normalise]
trace1_calibs_create_g          = [subframe, badpx_mask, bias]
trace2_calibs_create_g          = [subframe, badpx_mask, bias]
arc_calibs_create_g             = [subframe, badpx_mask, bias]
blazecor_calibs_create_g        = [subframe, badpx_mask, dark10.0, rflat, localbackground]
extract_calibs_create_g         = [subframe, badpx_mask, dark, rflat, localbackground]
wavelengthcal_calibs_create_g   = [subframe, badpx_mask]
standard_calibs_create          = []

subframe                        = [4096,2048,0,50]
subframe_nooverscan             = [4096,2048,0,0]
badpx_mask_filename             = /path/to/bad_px_mask.fits
background_width_multiplier     = [1.5, 0.5]
polynom_bck                     = [4, 3]
```

Figure 1: Example for setting up the calibration steps, in which darks are only available for some (long) exposure times and have been corrected with the overscan and a master bias was created elsewhere.
The first 10 lines show what calibration should be done for what types of data and the last 5 lines show parameters for some of the steps. As the master bias was already created, no calibration steps are applied to it. The overscan was removed in the dark frames (line 2), hence a different subframe is used.
To perform flat-field correction, the frames of the evenly illuminated detector are processed by cutting away the overscan area, the bad-pixel mask, and the master dark with the same exposure time (which is created following the settings in line 2). Before combining the frames using a median, the flat-filed frames are normalised using the median flux.
For extracting the science spectra (line 8) the subframe, the bad-pixel mask, the dark, the flat-field, and the background correction will be applied. For the flat-field (rflat) correction the image *master_rflat.fits* will be used, which itself will be loaded using *standard_calibs_create* (line 10) or is created by using the calibration given in line 3. Before extracting the spectra, the background flux (stray light) is removed from the reduced frame. The background image is created by fitting a 4 by 3 (cross-dispersion direction) 2-dimensional polynomial to the reduced frame, excluding the traces of the science orders (1.5 times the width) and the traces of the calibration orders (0.5 times the width).
A spectrum of the blaze function is extracted from the file created with parameters given in line 7. In this example the exposure time of the continuum source was 9.7 s, but no darks of 9.7 s exposure time were taken and therefore darks with 10 s exposure time should be used.
For the master files to search for the traces of the orders and the emission lines to create the wavelength solution no darks were available, hence the master bias will be used.

- Set parameter `original_master_wavelensolution_filename` to the file *master_wavelength.fits* in the previous night (contains the wavelength solution). In case no wavelength solution is required, set `master_wavelensolution_filename` to `pseudo`.

- When comparing the emission lines found in the current observation night with the original wavelength solution, several variations can be applied:

  order_offset It could be that not all or more echelle orders (especially on the red end) will be identified compared to the orders of the night, to which parameter `original_master_wavelensolution_filename` is pointing. The parameter `order_offset` gives the expected range of extra or fewer orders on the red side of the shift.

  px_offset If the setup is touched, the whole spectrum could be moved along the detector in dispersion direction, compared to the previous solution (given in parameter `original_master_wavelensolution_filename`). The parameter `px_offset` gives the expected range and step size of the pixel-shift.

  px_offset_order Gives the range and step size of a shift per order between the individual orders (e.g. the spectrum is tilted compared to the CCD alignment).

  resolution_offset_pct Gives the expected change of resolution in percent between the current setup and the previous wavelength solution. The pipeline will test 11 resolutions between 0% and the value given in parameter `resolution_offset_pct`.

- Run the scrip `prepare_file_list.py` to define what files should be used for what calibration.

- Run the scrip `reduction_day.py`.

4.3 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. The file must contain a 2-dimensional image with 1 for good and 0 for bad pixel.

bias : Apply a master bias. This can be either a pre-reduced image, or it can be created by the pipeline.

dark : Apply a master dark. A dark with the same exposure time as read from the fits header will be applied.

rflat : Apply a master true flat.

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 polynomial against the current reduced image, excluding the area of the science and calibration traces. The fitted 2d-image is then subtracted from the current image.

The following parameters change the way, several CCD images are combined. In the pipeline 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 be 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 as calibration step, 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`.

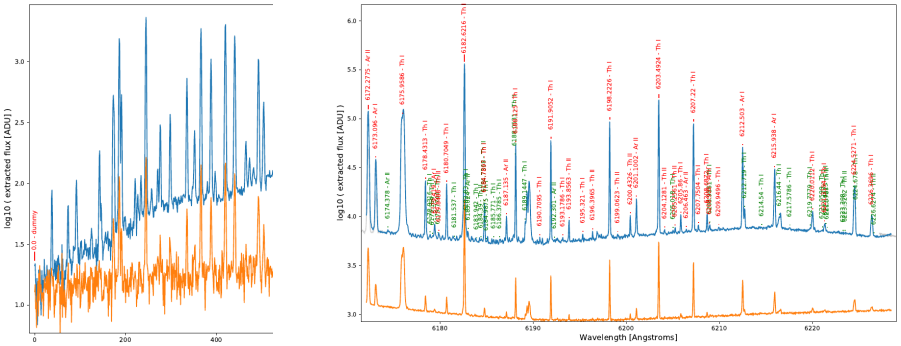
An example for the parameters with additional explanation is shown in Figure 1.

4.4 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_wavelensolution_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 perform most of the steps described in Section 5.1 and stops after searching for the lines in the emission line spectra with the message:

```
Error: Files for creating the wavelength solution do not exist ...
```

The order and pixel position of all identified lines, as well as the width and the height of the Gaussian fit, are stored in the file given in parameter `logging_found_arc_lines` (standard: *logging/arc_lines_found.txt*). This file can be opened (easiest in a spreadsheet application) and the corresponding wavelengths can be added to some of the lines (see Table 1). The lines for each order are sorted by the highest absolute flux value (this is not the height of the Gaussian fit) of the line,



(a) Part of a Th-Ar spectrum without a wavelength solution from a medium resolution spectrograph.

(b) Th-Ar spectrum for which the wavelength solution is known from a high-resolution spectrograph. The lines in red are the identified lines, green lines can be ignored.

Figure 2: The emission lines for a short (orange) and a long (blue) exposure time. The left figure covers about the same wavelength range as the Th-Ar spectrum with known wavelength solution in the right figure. The right image can be used to correlate pixel-positions and wavelengths in the left spectrum (see Table 1).

the saturated lines are excluded. Please note that sometimes the side-wing of a saturated line is listed in the found lines, these lines should be ignored as well.

The plots of the extracted spectra, which are stored in the file given in `logging_arc_line_identification_spectrum` (extended by the pipeline with *_manual.pdf*) can be used for visual guiding (for an example see Figure 2). Tests have shown that providing about 10 lines every 5 to 10 orders is enough for the pipeline to do the fitting in a good way. However, it is important that the lines cover as much of the CCD image as possible (e.g. covering most of the orders, inclusive the outermost lines of the orders). In case of a completely different setup an iterative approach might be necessary.

The correlated data needs to be saved into file *arc_lines_wavelength.txt* (in the working directory) with the following tab-separated entries (exactly one separator between each column):

- Aperture (starting at 0 for the reddest order)
- Real/physical order from the grating equation (bigger than 0, usually between 40 and 120, doesn't have to be exactly the right one). **Warn:** the difference between columns 1 (Aperture) and 2 (Real/physical) has to be the same for all lines.
- Pixel
- Wavelength [Å] (optional, can be also empty if the user wishes to copy the whole file from `logging_found_arc_lines` and only adds wavelengths to some lines)
- (optional: Type of the line (e.g. ThI, NeII)).
- (optional: Comments).

Table 1: Transforming the emission lines found by the pipeline into the file from which the pipeline will create the wavelength solution. A plot of the unknown and a reference spectrum can be found in Figure 2.

(a) Example of the entry of the file from parameter **logging_found_arc_lines** for one order.

22	244.85	1.02	2336.9
22	419.53	1.15	2085.9
22	365.86	1.09	1832.9
22	440.14	1.19	1571
22	186.32	0.93	1556.1
22	387.96	1.08	974.8
22	492.43	1.22	1013.3
22	334.94	1.06	658.1
22	173.30	1.54	505
22	503.78	1.23	368.8
22	298.19	1.04	316.3
22	276.35	0.95	247.6
22	351.78	1.09	197.2
22	532.12	1.06	108.5
22	464.67	1.14	123.7
22	142.64	0.95	161.1
22	233.87	0.87	56.7
22	91.77	1.16	80.8
22	397.14	1.18	47
22	257.30	1.15	32.4
22	37.76	1	76.2
22	73.44	0.99	50.2

(b) Example of *arc_lines_wavelength.txt* for one order as seen in a spreadsheet. Lines without a wavelength can, but don't have to be deleted. The text in the columns with line type and following are optional. Columns have to be spaced by exactly one tabulator in the exported txt-file.

22	47	244.85	6182.6216	ThI	
22	47	419.53		6212.719	blend, not used
22	47	365.86	6203.4924	Th I	
22	47	440.14	6215.938	Ar I	
22	47	186.32	6172.2775	Ar II	
22	47	387.96	6207.22	ThI	
22	47	492.43	6224.5271	ThI	faint blending
22	47	334.94	6198.2226	ThI	
22	47	173.30			outside comparison
22	47	503.78			
22	47	298.19	6191.9052		
22	47	276.35	6188.125		faint blending
22	47	351.78			
22	47	532.12			
22	47	464.67			
22	47	142.64			outside comparison
22	47	233.87	6180.7049	Th I	
22	47	397.14	6208.6872		
22	47	257.30	6184.7897		

Afterwards the script can run again and will use this data to create a new wavelength solution. Afterwards the output or logfile, and the file given in parameter `logging_arc_line_identification_spectrum` (standard: *logging/arc_line_identification_spectrum.pdf*) should be checked. The brightest lines of the reference file should be marked in red with the brightest emission lines of the spectrum (e.g. see Fig. 2b. If some areas of the spectrum (some orders, or the right or left side of some orders) are not fitted well, please add more lines of the problematic area to *arc_lines_wavelength.txt* and remove the following files (if standard names are used) before running the pipeline again:

```
rm master_wavelength*.fits
rm master_flat_spec_norm.fits
rm extracted/*
```

You might also want to read Chapter 8.3.

4.5 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 can be located in the `result_path` or `raw_data_path` (the first one has higher priority, in case different file exists in the two folders only the first file will be read). The file must contain one line per object. For each object the following tab-separated or comma-separated values need to be given (exactly one separator between each column):

1. Object name (the filename should start with the object name in order to be used)
2. RA (hh:mm:ss.ss or hh mm ss.ss or as one number in degrees)

3. DEC (±dd:mm:ss.ss or ±dd mm ss.ss or as one number in degrees)
4. PM RA (mas/year, can be 0 in the most cases)
5. PM DEC (mas/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. (optinal) Mask to use. At the moment G2, K5, or M2 are possible options. Can be empty.
9. (optinal) Velocity width to broaden the lines of the binary mask. Can be empty.

Please note that the position of Moon, Jupiter, and Sun are calculated by the pipeline using the mid-exposure time. This requires that the filename contains the (case-insensitive) object name. Therefore the object is not required in the object list.

4.6 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_blazecor: 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_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). Other formats can be defined in the procedure `get_obsdate`.

Table 2: 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	–

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 2. 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
- 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 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*.

Please note that the flag(s) need to be exactly one tabulator after the Observation time.

4.7 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.

4.8 Removing wrongly identified traces

After checking the results (see Chapter 5.6.1), 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 it's 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.

4.9 Finding further information - logged results

- 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. In the logfile, each entry will start similar to

  ```
  20190225160303 - 37448 -
  ```

  and decodes the current time (YYYYMMDDHHMMSS) of the entry and the PID[3] of the process that created it.

- 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`) in the a json[4] format.

- Images with the results of some steps can be found in a logging subfolder (standard: `logging`). Some of these images are shown in Figures 3 to 7.

- 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] process ID: https://en.wikipedia.org/wiki/Process_identifier
[4] https://en.wikipedia.org/wiki/JSON#Example

5 More information about how the pipeline works

5.1 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)).

   cal2_l : Long emission lamp exposure to create a good wavelength solution over the whole chip for the calibration fiber (standard: 5x 100 s ThAr or UNe).

   cal2_s : Short exposure in order to find the center of the saturated lines in cal2_l (standard: 5x 10 s ThAr or UNe taken between the cal2_l images).

   (cal1_l, cal1_s) : emission lamp spectra to create a wavelength solution for the science fiber (if applicable)

   blazecor : 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. 3). 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 (e.g. 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. 4).

4. Create the wavelength solution for the night on the calibration fiber spectra (and maybe the science fiber spectra). (→ Fig. 5 and 7). This is based on the solution of the previous data set, which is adjusted according to the given parameters (see Chapter: 4.2).

5. Extract the flat and normalise it. This is used for the blaze correction.

6a. (for single fiber spectrographs) Measure the instrumental drift in emission line spectra to calibrate the wavelength solution.

6b. (if applicable: find the radial velocity drift between the wavelength solutions of calibration and science fiber)

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) (Optional) Perform the radial velocity analysis (cross-correlation with template spectrum).
   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 create compatibility with other software.
8. (Optional) Perform the radial velocity analysis by building a template spectrum from the observation.

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. (Further information can be found in Chapter 8.)

5.2 Further settings for the parameters

The following steps give an overview about additional parameters which are not covered by Chapters 4.1 and 4.2.

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)

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`.

raw_data_exptim_keyword, raw_data_dateobs_keyword : Use the right header keywords for the exposure time and the observation date. The format for the observation date should be `%Y-%m-%dT%H:%M:%S.%f` or[5] `%Y-%m-%dT%H:%M:%S`. More formats can be added in the procedure *get_obsdate*.

[5] https://docs.python.org/2/library/datetime.html#strftime-and-strptime-behavior

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 Chapter 4.3 for options).

5.3 Necessary CCD images

To get the best results, the following data (or more files) should be taken. Please note that all filenames are case-insensitive. **No spaces or commas are allowed in the file names.**

- *(optional) 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**, **tung**, 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**, **Th_Ar**, or **UNe**. Results might be better, if the science fiber is dark.
- (*Only necessary, if the shift in dispersion direction between the fibers of a bifurcated fiber should be measured*): Arc2 (5x, calibration lamp through the science fiber, alternating a short (e.g. 5 s) and a long (e.g. 120 s) exposure time). The filename should start with **arc2**, **ThAr2**, **Th_Ar2**, or **UNe2**.
- (optinal) Bias (11x) and/or Darks (11x, for the exposure time of the true Flat and Flat)
- Science images: It is best to start the filename with the object name (as in reference file given by parameter `object_file`, see Chapter 4.5). However, in order to try to match the object name with entries in the `object_file` the parts of the filename containing "_" and "-" are stripped away one by one from the end of the filename.

The suggestions are for bifurcated fiber input to the spectrograph. If a spectrograph with a single fiber is used, then the same data should be taken using the same fiber (setting parameter `arcshift_side = center` will process the data correctly).

5.4 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 parameters `raw_data_dateobs_keyword` and `raw_data_exptim_keyword` (half of the exposure time, or a different fraction if one of the keys in `raw_data_mid_exposure_keys` exists). The pipeline can handle different standard formats (YYYY-MM-DDThh:mm:ss). If necessary, further formats can be added in the function *get_obsdate* in *procedures.py*.

- 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 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 in parameters `altitude`, `latitude`, and `longitude`.

- 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 Chaper 4.5).
  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, moon, or jupiter then the coordinates of the solar system are calculated for the mid of the observing time.

5.5 Format of the extracted files

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

5.5.1 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 polynomalial 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.

These files can be plotted with the pipeline. Please find instructions in Chapter 6.1.

5.5.2 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.

5.5.3 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.

5.5.4 raw-data filename + _lin_cont

This file contains data in the same form as described in Chapter 5.5.3, only that the continuum corrected data was linearised.

5.5.5 inside single folder

All the files are 2D arrays with the order/aperture as first and number of pixel in second dimension.

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

raw-data filename + _extr_bluefirst Same as the entry before, but starting with the blue orders instead of the red orders.

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

raw-data filename + _blaze_bluefirst Same as the entry before, but starting with the blue orders instead of the red orders.

raw-data filename + _wave The wavelength for each aperture and pixel.

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

5.5.6 path_csv_terra + <object name> + /data/YYYY-MM-DDHHMMSS.cvs files

The data in here is used in order to measure RV with Terra (https://drive.google.com/file/d/1xK-lYghFwpwtdXG9b4IbryYRd102q7So/view). The barycentric corrected wavelength solution and the continuum corrected spectrum (multiplied by the bad-pixel mask) are used. See Chapter 5.6.2 for more information.

5.6 Results from the pipeline

5.6.1 Preparing the data for one night

For each of the steps given in Chapter 5.1 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, trace, or arc_l.

Step 2: The trace of each scientific order is stored as a table in the file given in parameter `master_trace_sci_filename` (standard: *master_traces_sci.fits*). This is a table fits-file and 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`
- 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.

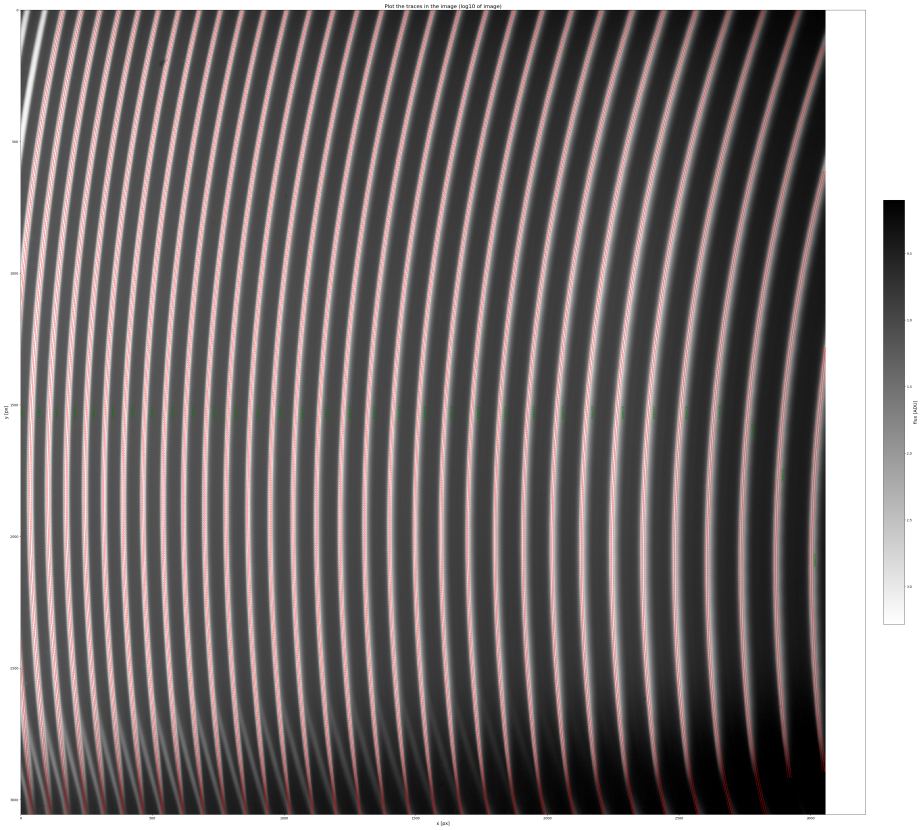


Figure 3: Reduced CCD image of a white light flat (log10 gray scale) with the marked traces of the identified scientific orders/apertures (red). The extraction width (determined from the width given in parameter **width_percentile** of the order multiplied with the value in parameter **extraction_width_multiplier**) is given in the dashed lines.

- 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. 3.

Step 3: The traces of the apertures of the calibration fiber are stored in the file given in parameter `master_trace_cal_filename` (standard: *master_traces_cal.fits*). This is a table fits-file and contains the same information as the result of **Step 2**, only that the lowest order of the parameters of the polynomial fits are 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: *arc-traces_in_master_traces_cal.png*). An example is show in Fig. 4.

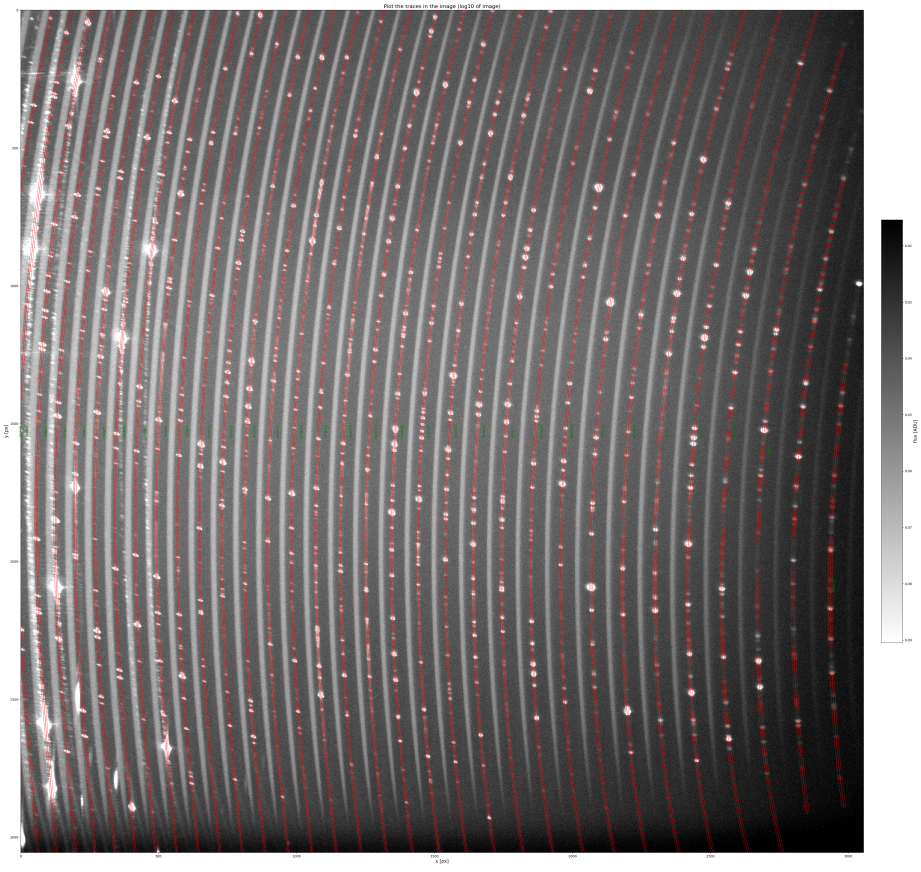


Figure 4: Reduced CCD image of a ThAr exposure (log10 gray scale) with the marked traces of the identified calibration orders/apertures (red). The extraction width (determined from the width given in parameter **width_percentile** of the order multiplied with the value in parameter **extraction_width_multiplier**) is given in the dashed lines.

Step 4: The wavelength solution is stored in a table fits-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 dir

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 $y = (m_0 + \tilde{m})\lambda_c$, 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

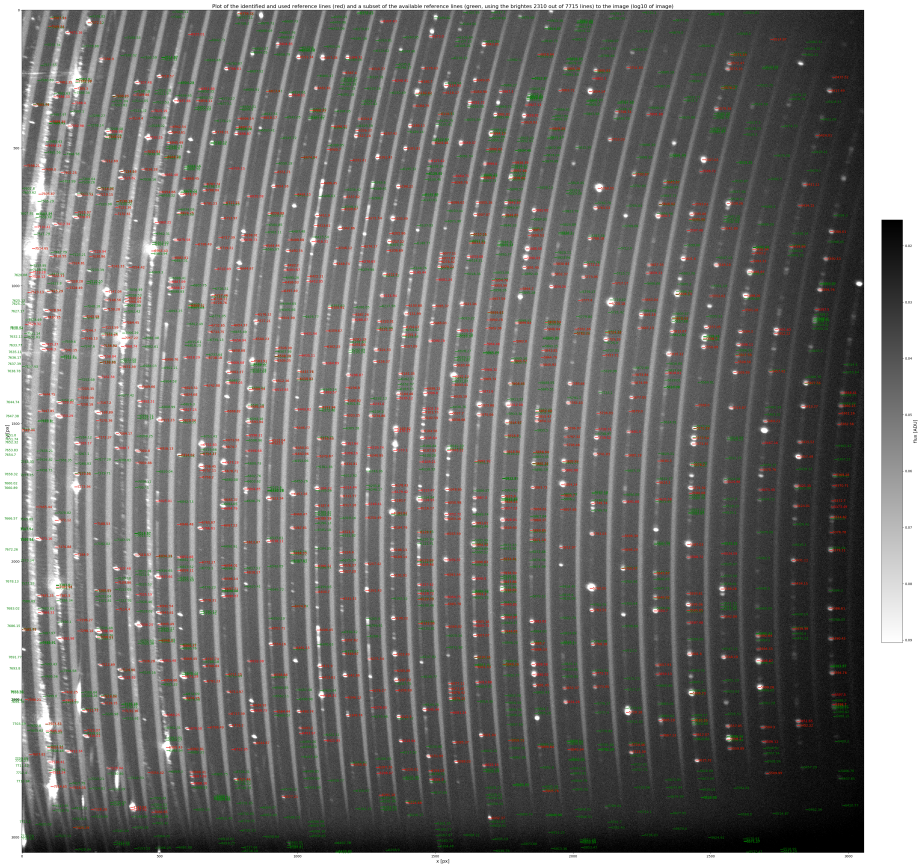


Figure 5: Reduced CCD image of a ThAr exposure (log10 gray scale) with the identified lines from the reference catalogue (red). Only this set of data was used to create the wavelength solution. The remaining lines of the reference catalogue, which weren't used for fitting the solution are shown in green. The data on an order by order basis is shown in Figure 6.

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 Chapter 5.5.1

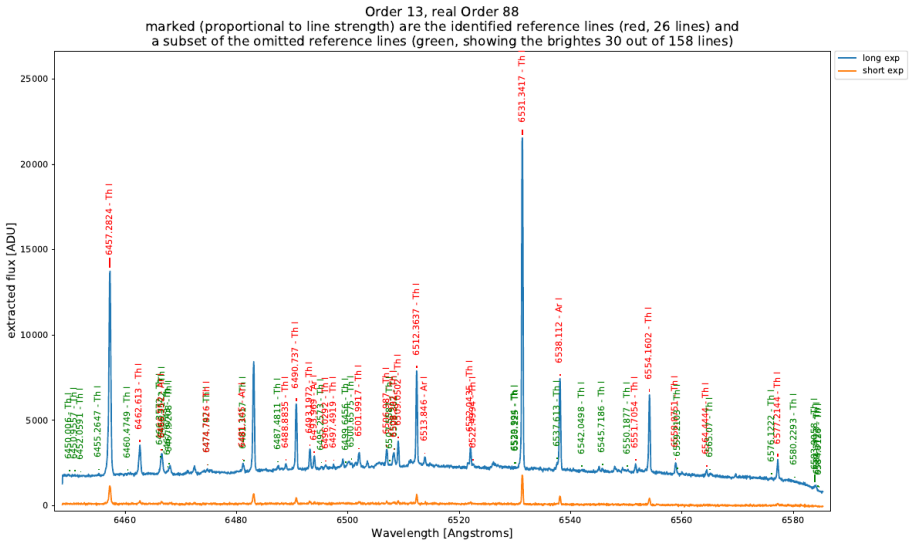


Figure 6: One page (aperture 13, real order 88) of the spectral atlas (file created from parameter **logging_arc_line_identification_spectrum**) created from the wavelength solution. Blue shows the spectrum of the emission lines in the long and orange the emission lines in the short exposure. The identified 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), for better wavelength solutions the line catalogue should be cleared of bad lines.

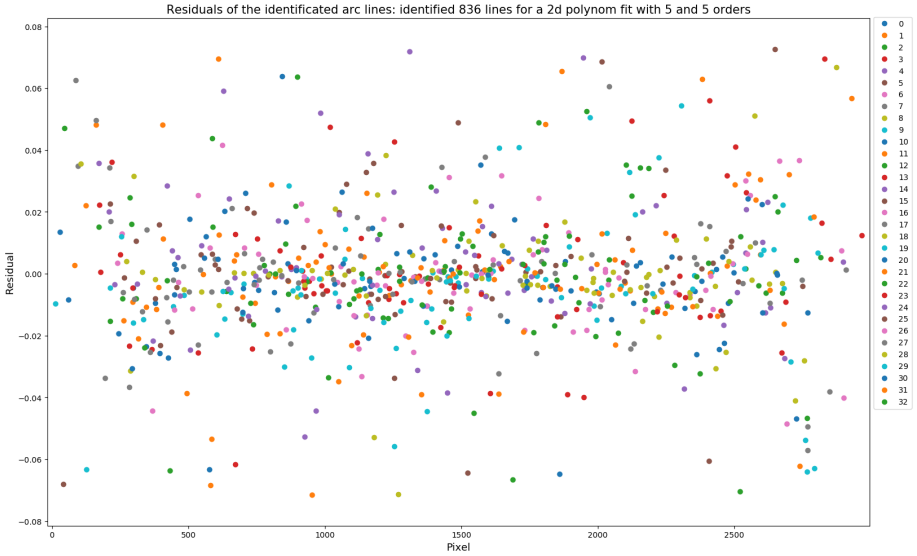


Figure 7: Residuals between the identified catalogue lines and the wavelength solution. Colorcoded are the different apertures.

5.6.2 Extracting the science data and radial velocity analysis

Files in folder given in parameter path_rv_ceres (standard: ceres_rv_output/ This step is only done, if the Ceres pipeline is installed and `path_ceres` is set correctly to the base folder of Ceres and the sub-folders *utils/Correlation*, *utils/GLOBALutils*, *utils/OptExtract*, and *utils/CCF* exist.

***_<mask>.pdf:** Cross-correlation function between object spectrum and the template.

pkl-files: Data used to plot the cross-correlation function in the python-pickle format.

stellar_pars.txt-files: Stellar parameters of the object: T_{eff}, $\log g$, Z, $v \sin i$, *vel*0

Files and folders in folder given in parameter path_csv_terra (standard: terra_rv_data The data of the individual objects are stored in this folder in the subfolder *object name + /data/*. If Terra is installed and `terra_jar_file` is set correctly to the Terra *PRV.jar*-file, then Terra will be run on each *object name* within the folder. The results of the radial velocity analysis will be stored in *object name + /results/synthetic.rv*.

It is possible to include the information from previous nights of the same object by linking or copying the *.csv*-files into the corresponding *data/*-folders and rerunning `python <path to scripts>/reduction_day.py`. Please note that in order to this, the number of orders and the number of extracted pixel has to be always the same.

5.7 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 Chapter 4.1 and 4.2).

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.

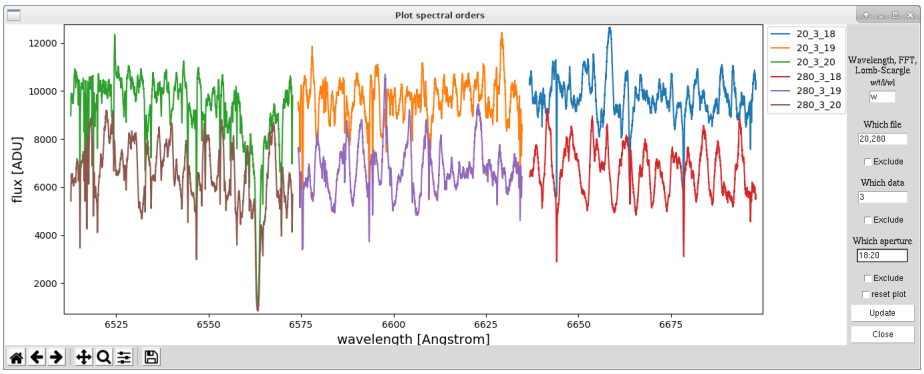


Figure 8: An example of how to plot the extracted data. Shown are the solar spectra of entry 20 and 280 in the plot_files.lst file (*Which file*). The graph shows the blaze corrected flux (*Which data*: 3), plotted over the wavelength (w). Only apertures 18, 19, and 20 are shown. Each of the selections can be inverted by ticking the *Exclude*-boxes below their entries.

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 reads the files as described in Chapter 5.5.1 and which are listed in the file `plot_files.lst`. At the first start it will plot the extracted spectrum for all orders of all files in the list. The spectra will be plotted in the pixel-reference.

This plot uses the matplotlib libraries and a toolbar for interactive navigation is available[6] in the lower left corner. Please refer to Footnote 6 for instruction on the use. While the changes done with the toolbar are instant, the changes of the parameters on the right have to be updated using the button provided. To reset the plot to automatic scaling enable *reset plot*. Please be aware that once the window size has been changed and the plot is updated again the margins will be adapted to use the plot area in an optimal way.

Further limitations of the plotted data can be done using the GUI. For a plot in different x-axis the first text box can be used. See the list in Table 3 for options. For plotting only data from a subsection of files, the first text boxes *Which file* and *Exclude* below this field can 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 box are used (see an example in Fig. 8 and 9).

[6] https://matplotlib.org/3.1.1/users/navigation_toolbar.html

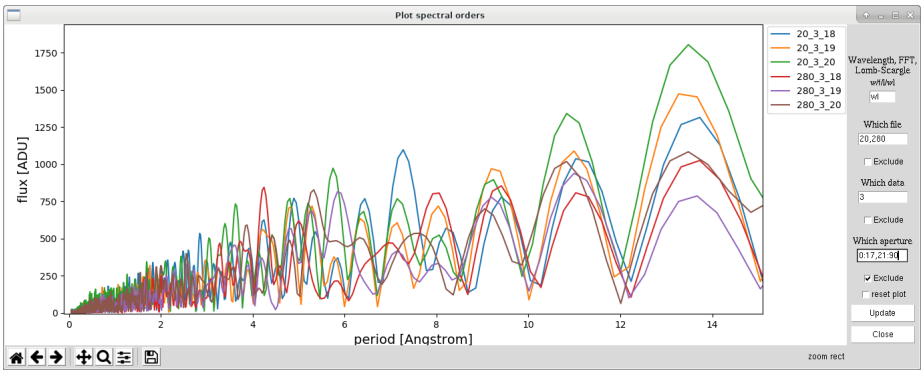


Figure 9: Another example of how to plot the extracted data. Shown is the Lomb-Scargle Periodogram (wl) for the same spectra as shown in Figure 8 of files 20 and 280 in the plot_files.lst file. The blaze corrected spectra (data type 3) of all apertures except 0 to 17 and 21 to 280 (this means only apertures 18 to 20) and their corresponding wavelength solution was used as basis for the calculations.

Table 3: Options for the data on the x-axis for plotting.

(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.

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. Check only 'HARPS/LaSilla'. For example data select night '2015 07 30' to find data of the asteroid 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_wavelensolution_filename = master_wavelensolution_manual.fits`. The file `pixel_to_wavelength.txt` needs to be created (Chapter 4.4).
- `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_blazecor = 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`
- `blazecor_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.

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. 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[7]:

- 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 -¿ 'ccf' and 'bis' files
- correction for instrumental drift using the calibration fiber

The following files are available:

[7] http://www.eso.org/rm/api/v1/public/releaseDescriptions/72

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.

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[8]. 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).

[8] See page 22 in https://www.eso.org/sci/facilities/lasilla/instruments/harps/doc/DRS.pdf

7.1.5 Analysing the reduced HARPS data with TERRA

For more information about TERRA please refer to the literature[9]. 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

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 = [1201,511,380,0]`
- `rotate_frame = 270`
- `maxshift = 5`
- `bin_search_apertures = 10,1`
- `bin_adjust_apertures = 2,1`
- `width_percentile = 5`
- `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,bias`
- `trace1_calibs_create_g = subframe,bias`
- `trace2_calibs_create_g = subframe,bias`
- `extract_calibs_create_g = subframe,bias`
- `blazecor_calibs_create_g = subframe,bias`
- `wavelengthcal_calibs_create_g = subframe,bias`
- `site = TNT`
- `altitude = 2457`
- `latitude = 18.59055`
- `longitude = 98.48655`
- `extracted_bitpix = -32`

[9] https://ui.adsabs.harvard.edu/#abs/2012ApJS..200...15A/abstract

8 Fixing problems: The results are not as expected

In the following Chapters a few ideas of how to check and fix common problems are given. The following standard names are assumed.

- `object_file = object_list.txt`
- `master_trace_sci_filename = master_traces_sci.fits`
- `master_trace_cal_filename = master_traces_cal.fits`
- `master_wavelensolution_filename = master_wavelength.fits`
- `logging_path = logging/`
- `logging_trace1_binned = mstr_trace1_binned.fits`
- `logging_traces_binned = sci_trace1_binned.fits`
- `logging_traces_im_binned = traces_in_master_trace1_binned.png`
- `logging_traces_im = traces_in_master_trace1.png`
- `logging_find_arc_traces = arctraces_find.png`
- `logging_arctraces_im = arctraces_in_master_trace2.png`
- `logging_arc_line_identification_residuals = arc_line_identification_residuals.png`
- `logging_arc_line_identification_spectrum = arc_line_identification_spectrum.pdf`
- `logging_arc_line_identification_positions = arc_line_identification_positions.png`

If you modified the parameters, please use yours instead of the given ones in the following Chapters.

If parameters are changed while working through the following steps. Some of the already created files need to be removed before the pipeline can re-run with the better settings. In the following chapters, each correction step should name the first value of the following list, that need to be removed (error messages because of missing files can be ignored). Afterwards `reduction_day.py` can be run again.

```
rm master_*
rm logging/sci_trace1_binned.fits
rm master_traces_sci.fits
rm master_traces_cal.fits
rm master_wavelength*.fits
rm master_flat_spec_norm.fits
rm extracted/*
```

8.1 Are the traces (of the science fiber) identified correctly?

- **logging/traces_in_master_trace1.png**: Are the orders identified correctly? If yes, go to next chapter.
- If a lot of unexposed area is located on the image borders, adjust parameter `subframe` to the correct image set. Remove everything after *master_*.
- If the orders are not running from up to down in the image, the adjust paramter `rotate_frame`. Remove everything after *master_*.
- If noise was marked as order, run `remove_orders.py` and remove the wrong orders. All depending calculations will run again.

- If orders are missing check **logging/traces_in_master_trace1_binned.png**. If the orders are marked there correctly then adjust parameter `bin_adjust_apertures`, so that the echelle orders are well separated and only pixel with similar amount of flux are median combined. Remove everything after *master_traces_sci.fits*.

- If orders are missing check **logging/mstr_trace1_binned.fits**. Very likely, too heavy binning was used and the orders are not destinguishable or half of the binning area consists of background pixel. Adjust parameter `bin_search_apertures`. Remove everything after *logging/sci_trace1_binned.fits*.

8.2 Are the traces (of the calibration fiber) identified correctly?

- **logging/arctraces_in_master_trace2.png**: Are the orders identified correctly? If yes, go to next chapter.

- Be sure, that all echelle orders from the science fiber are well separated from the orders of the calibration fiber. If necessary change the orientation of the bifurcated fiber in the setup (angel between bifurcated fiber and cross-disperser). Take new data, start in a new folder.

8.3 Is the wavelength solution correct?

- **logging/arctraces_in_master_trace2.png** (if emission line lamp spectrum, otherwise **logging/arc_line_identification_positions.png** (ignore the written wavelengths)): If the wavelengths of the emission lines is not increasing from up to down along the individual orders or if the redder orders are not on the left side of the image, adjust parameters `rotate_frame` and `flip_frame`. Remove everything after *logging/sci_trace1_binned.fits*.

- **logging/arc_line_identification_residuals.png**: If the most dots fall on the x-axis , go to next chapter. If most dots fall on a sinusoidal line, increase the number of orders in parameters `polynom_order_traces` or `polynom_order_intertraces`. Remove everything after *master_wavelength*.fits*.

- **logfile**: search for the (last) line containing:

  ```
  standard deviation of the residuals between the lines and the fit
  ```

  A high number of lines should have been identified to a reasonable precision (compare to a previous night). If all is the case, go to next chapter.

- **logging/arc_line_identification_spectrum.pdf**: The emision lines in the extracted spectrum should match with the brightest lines from the catalogue (red wavelengths, marker lengths is an indication for the brightness). If yes, go to next chapter.

- The lines in the above spectrum are not matching at the borders. Decrease or increase parameter `polynom_order_traces`. Decrease or increase the values in `opt_px_range`. Remove everything after *master_wavelength*.fits*.

- The lines are completely off. Compare **logging/arctraces_in_master_trace2.png** in the current directory and in the directory from where the the original wavelength solution are taken (`original_master_wavelensolution_filename`). Check, that the parameters `order_offset`, `px_offset_order`, and `px_offset_order` are large enough to cover the change between the two nights. Be sure the values are not too big. Remove everything after *master_wavelength*.fits*.

8.4 Are the object and telescope coordinates correct, is the handling of time right?

- Was the right object used from **object_list.txt**?
- Compare the barycentric velocity with other calculators. Are the object and telescope coordinates right? Is the time zone correct? If not, check parameters `raw_data_dateobs_keyword`, `raw_data_timezone_cor`, `site`, `latitude`, and `longitude`.

8.5 Error message: "urllib2.URLError: <urlopen error timed out>"

When the barycentric correction are done, the python package *barycorrpy* is downloading data from the internet, e.g. for the leap second management. It also calls the package *astropy*, which downloads position files. If the internet connection is not available during the download the pipeline might fail. In this case try to re-run the pipeline again, it might have been just a slip in the internet connection. If it continuously fails you can check that following resources are available (e.g. in you browser):

- http://www.astropy.org/astropy-data/coordinates/sites.json
- http://maia.usno.navy.mil/ser7/finals2000A.all
- https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/a_old_versions/de405.bsp
- http://maia.usno.navy.mil/ser7/tai-utc.dat

8.6 Further help

If you observe strange behaviour, please contact Ronny[10] with a description of the problem and the error message. Access to the reduction folder would be preferable.

[10] r.errmann@herts.ac.uk

9 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 B).

9.1 With Anaconda and python 2.7

Anaconda can be downloaded from https://www.anaconda.com/distribution/#linux. 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` (check with `echo $0` what environment you are using) 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 `tcsh` 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`).

9.1.1 Additional modules

Once Anaconda is installed few 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 barycorrpy
pip install ephem
```

9.1.2 Manual installation of barycorrpy

In case *barycorrpy* is not available using *pip*, please install it by using the instruction of the package: https://github.com/shbhuk/barycorrpy/wiki/1.-Installation Afterwards, please make sure that `import barycorrpy` works in python.

9.1.3 Install pyfits for radial velocity analysis

To use the cross-correlation based radial velocity from the CERES pipeline (see Section 9.4.1) 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
```

9.1.4 Upgrading numpy for radial velocity analysis

To use the cross-correlation based radial velocity from the CERES pipeline (see Section 9.4.3) the package numpy needs to be on version 1.13 or newer. To check please run:

```
pip show numpy
```

Numpy can be upgraded in anaconda using:

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

If the version after the upgrade hasn't change, it might be necessary to run

```
pip uninstall numpy
conda update numpy
```

9.2 With Anaconda and python 3.7 (in preparation)

Next thing to https://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html#returning-iterable-objects-instead-of-lists, procedures.py after line 2077 needs be done.
Anaconda can be downloaded from https://www.anaconda.com/distribution/#linux. Please select the version for python 3.7. After download it can be installed by running

```
sh Downloads/Anaconda3-2018.12-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` (check with `echo $0` what environment you are using) add the following line (replace */home/exohspec* by the path you used for the installation to your `.bashrc`.

```
export PATH="/home/exohspec/anaconda3/bin:$PATH"
```

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

```
setenv PATH /home/exohspec/anaconda3/bin\:$PATH
```

Please make also sure that the environmental variable `PYTHONPATH` does not point to any python-2.7 folders by checking `echo $PYTHONPATH` . If python-2.7 folders appear, set the `PYTHONPATH` variable only to python-3 folders using

```
export PYTHONPATH="<subsection of PYTHONPATH>"
```

in a bash environment or

```
setenv PYTHONPATH <subsection of PYTHONPATH>
```

in a c-shell environment.

After making the change and opening a new terminal, python should point to the file in the Anaconda installation (`which python`) and the path should be set correctly (`echo $PYTHONPATH`).

9.2.1 Additional modules

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

```
pip install gatspy
pip install barycorrpy
pip install ephem
```

9.3 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/v0.4.0.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 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).

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.

9.4 External packages for Radial velocity analysis (optional)

The pipeline creates extracted spectra in different formats, which then can be further analysed using available software. Additionally the pipeline can use available packages to find the radial velocity of the object. Below are described the requirements of the external software.

9.4.1 Ceres Pipeline

Authors: Rafael Brahm, Andrés Jordán, Néstor Espinoza. If you make use of the Radial Velocity analysis from the Ceres , please cite Brahm et al. 2017[11].

The Package can be downloaded from
https://github.com/rabrahm/ceres ("Clone or Download"-button)
and then unpack the archive. To make it work, please follow the Installation Chapter in the README.md file on
https://github.com/rabrahm/ceres .

9.4.2 Terra

Terra can be downloaded from
https://drive.google.com/file/d/1xK-lYghFwpwtdXG9b4IbryYRd102q7So/view . It only has to be extracted. To check that all dependencies are installed on the system one can run.

```
java -jar <path to terra>/terra/PRV.jar
```

This should produce the entry **** TERRA v1.8 **** before failing.

9.4.3 Serval

To install Serval please follow the instructions as given on
https://github.com/mzechmeister/serval . It is worth to check that the software is working by running the package on the provided test data. Serval only works with python 2.7.

[11] https://ui.adsabs.harvard.edu/#abs/2017PASP..129c4002B/abstract

A Changelog (old)

Please found the changes of newer versions in Chapter 3.

v0.3.3

- Added parameter `raw_data_timezone_cor` to conf.txt.

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.
- `extraction_width_multiplier = 1`, `arcextraction_width_multiplier = 1` `width_percentile = 05`, and `extracted_bitpix = -32` are standard now.

B Module dependencies if not using Anaconda (needs update)

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

C Short introduction to reduction of CCD images

For more information please refer to the literature, e.g. Steve Howell: Handbook of CCD Astronomy (Chapter 3).

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.