User Manual for Data Reduction using the HiFLEx package
Working version: v1.0.0

Ronny Errmann, EXOhSPEC team
Physics, Astronomy and Maths, University of Hertfordshire

May, 2020

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 this package, please cite Errmann, et al. 2020[1] and let me know.

If you use the barycentric corrected radial velocities, the barycentric velocity, or the BJDTDB, please cite Kanodia and Wright 2018[2].

If you use the radial velocities from TERRA, SERVAL, or CERES please cite the corresponding papers: Anglada-Escudé et al. 2012[3], Zechmeister et al. 2018[4], or Brahm et al. 2017[5].

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)[6] 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 Figure 6 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://adsabs.harvard.edu/abs/2020PASP..132f4504E
[2] https://iopscience.iop.org/article/10.3847/2515-5172/aaa4b7
[3] https://adsabs.harvard.edu/abs/2012ApJS..200...15A
[4] http://adsabs.harvard.edu/abs/2018A&A...609A..12Z
[5] https://adsabs.harvard.edu/abs/2017PASP..129c4002B
[6] https://www.cambridge.org/core/books/handbook-of-ccd-astronomy/97D3D910788D44D11394B3B57C3FA743

3 Changelog (recently)

Please find the changes of older versions in Chapter A.

v1.2.0

- Added parameters `pxshift_between_wavesolutions`, `dataset_rv_analysis`, `logging_arc_line_identification_residuals_hist`, `logging_em_lines_bisector`, `logging_blaze_spec`
- Python 3 ready (Pipeline will also keep working under Python 2)
- Blaze correction with a fitted function (polynomial of user defined order)
- Extra logging information: Distribution of the residuals for the wavelength solution, graphs for measurements offsets to the wavelength solution, Bisector of the emission lines used for the wavelength solution

v1.1.0

- Improved measurement of the offset between emission line spectra and the wavelength solution
- Small improvements in the code

v1.0.0

- Rename of `prepare_file_list.py` to `file_assignment.py`
- Rename of `reduction_day.py` to `hiflex.py`
- Added parameters `logging_traces_bisector`, `extraction_shift`, `extraction_min_ident_part_of_trace_percent`, `extraction_precision`, `sigmaclip_spectrum`
- Added parameter `path_rv_serval`
- Renamed parameter `path_csv_terra` to `path_rv_terra`
- Added parameters `max_cores_used_pct`, `dataset_rv_analysis`, `blazercor_function`
- Added SERVAL radial velocity analysis
- Bugfixing CERES RVs
- Format of the object list was changed to keep compatibility with CERES. To convert run `transform_coordinate_file_before_v1.0.0_202004.py` before running any other Script.
- Improved code, general bugfixes, aditional features

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_paths` 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 4.3). Otherwise leave empty or 'NA'.
   - Adjust `rotate_frame` and `flip_frame` so that the orders are up to down (blue wavelength on top) and red orders are on the left. The pipeline will first rotate and then flip, the flip is swapping left and right (Fig. 6 shows the orientation).
   - Adjust parameter `subframe` to a subframe, if only part of the CCD should be used. Otherwise leave empty or set to the full detector size.
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` (left or right). Set it to center for a single fiber spectrograph.
4. Create a new wavelength solution (in case no wavelength solution is required, set `original_master_wavelensolutio` to `pseudo` and with the next numbered 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 can be given optionally (called *pixel_to_wavelength.txt*, Chapter 4.7).
   - 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 configuration file shows the settings for data taken with MaximDL and for data from the HARPS spectrograph. See also 4.5.
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 4.4 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 4.12).
8. For RV analysis: Set path `terra_jar_file` to the TERRA *PRV.jar* file. Set paths `path_serval` and `path_ceres` to the home folders of the SERVAL and CERES pipelines. See also in 9.4.
9. Run the scrip `file_assignment.py` (see 4.5 below for more information):
   - Define what files should be used for what calibration.
   - Define what files to extract (and in which RVs will be measured).
   - Please note that you can use already reduced images. In this case the reduction steps as defined in `conf.txt` 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 `hiflex.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 (see4.1), 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_filenam` 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`.

- Activate the hiflex environment in anaconda (if set, see 9.5.1) `conda activate hiflex` .
- Run the scrip `file_assignment.py` to define what files should be used for what calibration.
- Run the scrip `hiflex.py`.

4.3 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.

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. 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 : For Scattered light removal. 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. The master files for bias, rflat, and darks can also be copied into the result folder from a previous night.

4.5 Assigning the observed data and calibration data to the pipeline

The script `file_assignment.py` reads all files in the folders (and subfolders) given in parameter `raw_data_paths`. 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_blaze: 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`. If the time zone is not UT, then `raw_data_timezone_cor` can be used for correction, e.g -7 when local time in Thailand has been stored as UT time.
If the information of exposure metre to calculate the weight of the mid exposure time, a list of header keywords can be given in `raw_data_mid_exposure_keys`.

raw_data_object_name_keys: List of header keywords to get the object name, which is then used to derive the information from Simbad.

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 (*file_list_raw_data.txt*, which is shown to the user in a GUI (see Figure 2).

- Comment the file out (don't use it)
- Observation time
- Exposure time in seconds
- Filename
- What calibration should the file be used for:

  Bias: Use it to create a master bias.

  Dark: Use it to create a master dark with this exposure time.

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

condition	fiber1	fiber2
filename contains `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	–

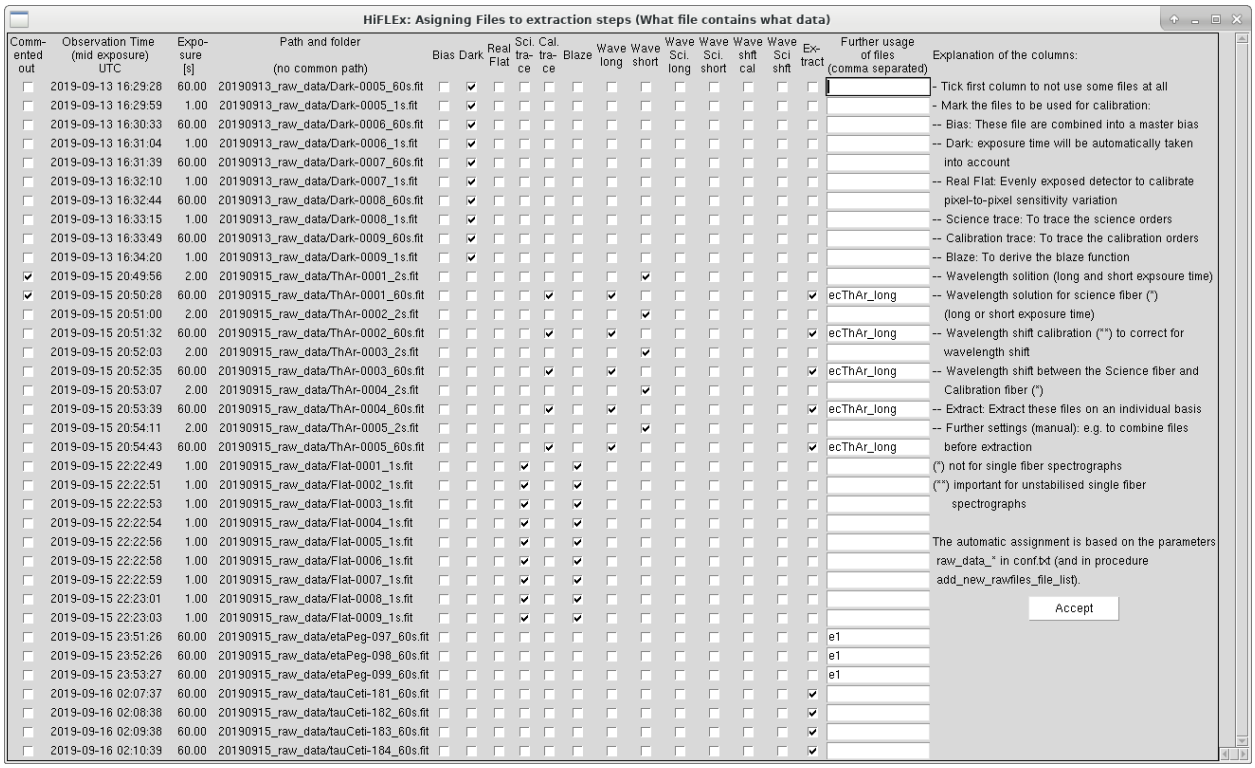

Figure 2: User interface to select which files should be used for which calibration step. In this selection the Dark-files will be used for dark correction, the Flat-files (Tungsten lamp) will be used to find the science traces and to find the blaze function, the long-exposed ThAr will be used to find the traces of the calibration fiber and for creating the wavelength solution for the calibration fiber, the short-exposed ThAr will also be used for creating the wavelength solution (for lines saturated in the long exposure). The spectra of the stars will only used to be extracted, together with the long-exposed ThAr individually and the combined ThAr with the long exposures (*ecThAr_long*). The first two ThAr files will not be used. Please note, in this data set no data to create a wavelength solution in the science fiber is available.

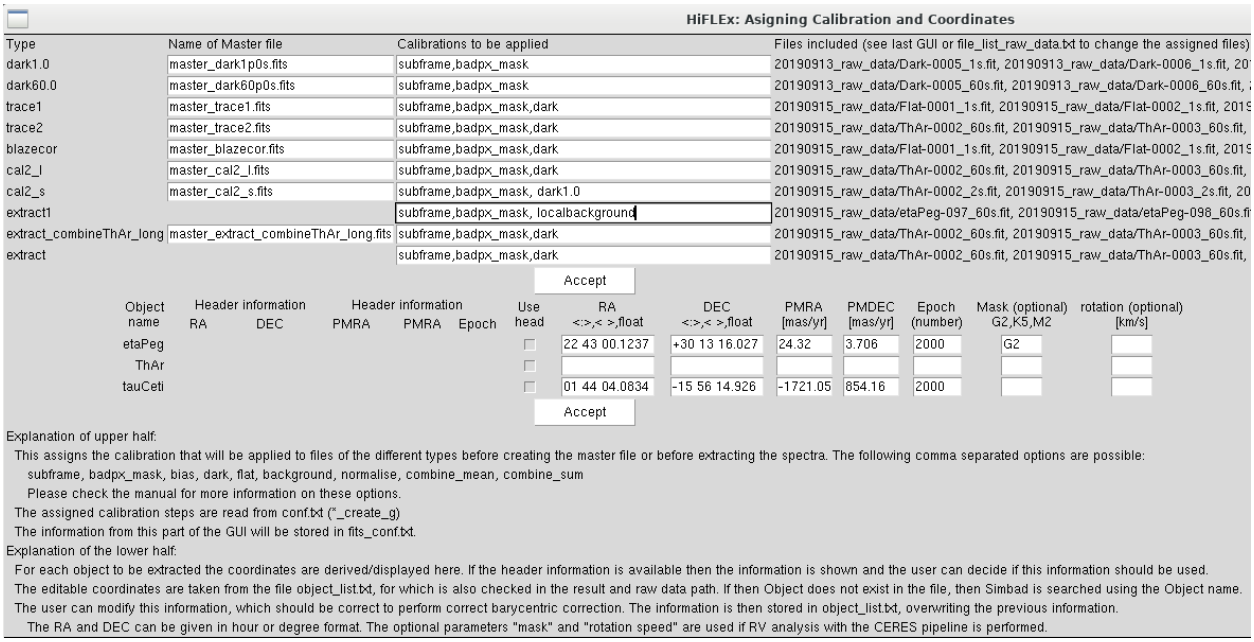

Figure 3: The second step for preparing the files using the data as shown in Figure 2. The upper half shows all the calibration sets and what processing of the files to be done. The exposure time of the ThAr for *cal2_s* was 2 seconds, however no darks were available. Hence the user assigned the darks with 1 second exposure time. The files for eta Peg were put under a different extraction step, as the user wanted to replace *dark* correction with *background* correction.
No header information about the objects was available, but the lookup of the object name on Simbad was successful.

Real flat: For master full flat (detector evenly illuminated).

Sci. trace, Cal. trace: To find the traces of the science and calibration fiber (for bifurcated fiber input).

Blaze: Use the file to create a master file from which the blaze function is extracted.

Wave long, Wave short: Files from which the wavelength solution for the calibration fiber will be created. The files will first combined into a master file and then extracted. In case of single fiber fed spectrographs, these settings should be used.

Wave Sci. long, short: For bifucated fiber fed spectrographs to create the wavelength solution for the science fiber.

Wave shft cal, Wave Sci shft: Define the files from which a time dependent drift between the wavelength solution is calculated. If ThAr light is used in both fibers, than both calibrations should be ticked. The script will calculate the shift using the pairs closest in time. For single fiber fed spectrographs only "Wave shft cal" is used to derive the shift from the wavelength solution.

Extract: Extract these files individually. The processing as given in parameter `extract_calibs_create_g` will be assigned.

Further entries: Further ways to extract the science spectra is possible. Several options can be combined using comma:

e <obj >: Extract files individually. The processing as given in parameter `extract<obj>_calibs_create_g` will be assigned. If this parameter doesn't exist, then parameter `extract_calibs_create_g` will be used.

ec <obj >: The same as **e <obj >**, but instead of extracting each file individually, all files with **ec <obj >** will be combined into a file called `master_<obj>.fits` and then this file is extracted.

If the automatic assignment of the the calibrations is wrong, please check the parameters `raw_data_*` in conf.txt. If this doesn't fix it, please check the hard coded lines in procedure *add_new_rawfiles_file_list* as some of the assignments had to be hard coded.

After the assignment which files to be used for which calibration the data reduction steps for each of these calibrations are shown in a GUI (see Figure 3). Thereby the information are read from the configuration file from section `*_calibs_create_g`. The describtion of the parameters can be found in 4.4.

The header information about the coordinates of the object are shown, if available. Furthermore, the information stored in the file given in parameter `object_file` are read, and if the file does not exist or the file does not contain the object, the object name is looked up on Simbad. Please note that the position of Moon, Jupiter, and Sun are calculated by the pipeline using the mid-exposure time if the object name can be derived from the header or the filename. Therefore the object is not required in the object list.
When Accepting the settings from the GUI, the data will be stored in the files given in parameters `configfile_fitsfiles` and `object_file`. Both files can be edited, and further modification of the object coordinates is described in 4.13.

4.6 Removing wrongly identified traces

After checking the results (see 4.15.1), 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 its high 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 `hiflex.py` will run again.

4.7 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 4.9 and then open a GUI similar to the one shown in Figure 4. The user can also prepare the correlated data between pixel and wavelength beforehand, which is described below the GUI instructions.

To create a new wavelength solution the following steps are suggested:

1. Find an order close to the central order with at least 2 prominent lines for which the wavelength is know, best several 100 or 1000 pixel apart of each other by changing the order number and *Update*.
2. Add the wavelength for these two lines in the table.
3. *Update* the plot to see the linear fit of the catalogue wavelength in blue in the plot.
4. Add a few more wavelength using the brightest blue lines (longest marker), which should be close to the position of the brightest emission lines. Please note that the spectrum might deviate from the linear fit. Please also note that for most lamps the gas (Ar) lines are brighter than the metal (Th) lines, however the catalogue might not represent this ratio.
5. Repeat steps 1 to 4 for a few more orders. If consecutive orders are overlapping in wavelength some of the lines could be easily identified in the neighbouring orders.
6. Set the order offset to a value about the expected value. Set the polynom-orders to 2. *Calculate wavelength solution*.
7. Scan through the orders and add few more wavelengths when the fitted positions of the catalogue lines (green lines in the centre of the central plot) deviate from the emission lines. *Calculate wavelength solution*.
8. Repeat the step 7 until the lowest and highest orders have a good wavelength solution. It might be necessary to increase the polynom-orders.
9. Set the polynom-orders to the final values (e.g. 4 and 4), *Calculate wavelength solution* and *Accept*.

Prepared correlated data between pixel and wavelength: The user can also prepare the correlated data between pixel and wavelength. It needs to be saved into file *pixel_to_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).
- 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: Further entries (e.g. ThI, NeII, or comments)).

You might also want to read Chapter 8.3.

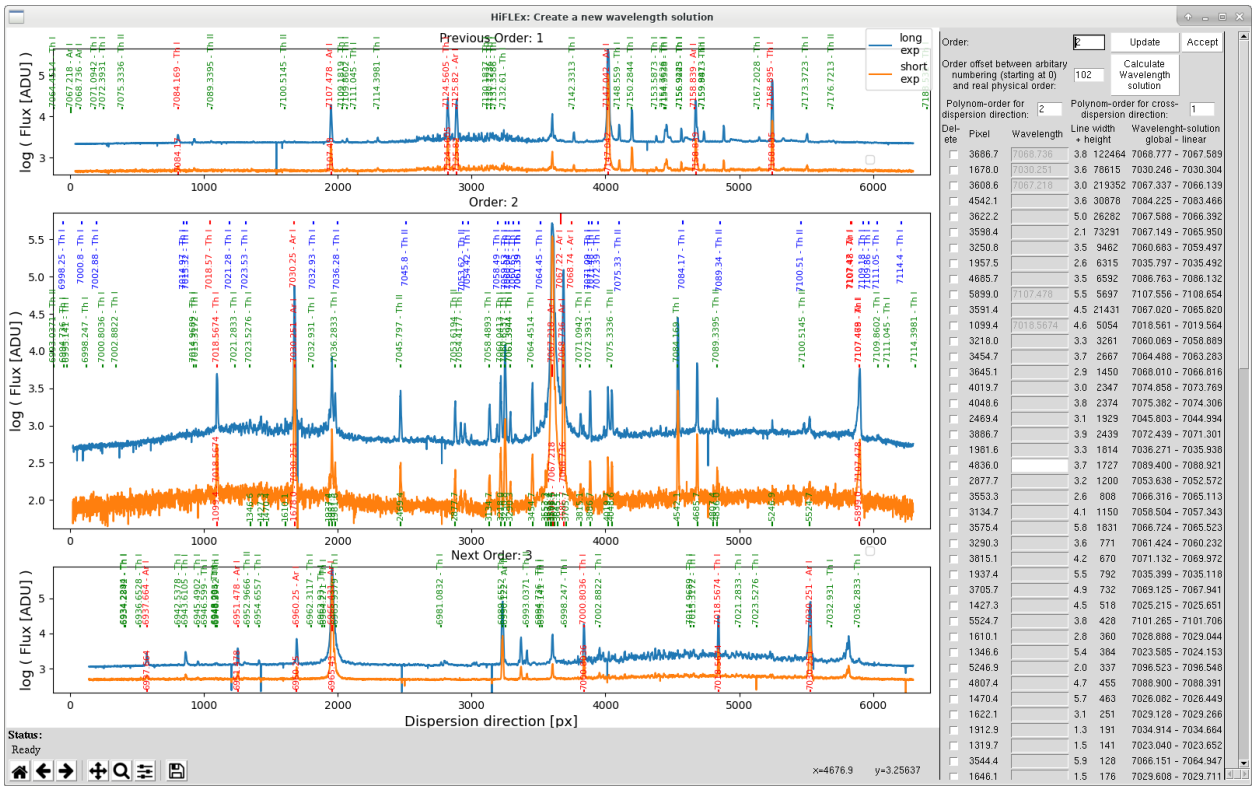

Figure 4: Window to create a new wavelength solution showing short and long exposed emission line spectrum of the current selected order and the previous and next order above and below, respectively. The bottom of the central plot shows the identified lines in pixel and, if a wavelength was given, this wavelength. In the other plots only the wavelength is shown. Red shows lines for which a wavelength was provided. The list on the right show the pixel positions of the identified lines, ordered by peak intensity.
In this example the user clicked on the line around pixel 4836, this entry is then selected in the list on the right to add the wavelength. Clicking away from a line will make all wavelengths editable again.
To update the plot or to change the order, the button *Update* can be used. If at least two wavelengths are given for the current order, a linear fit will be performed. The lines from the emission line catalogue (parameter *reference_catalog*) are shown for this fit in red or blue (wavelength given or not) at the top of the central plot. The marker gives the relative line strength from the catalogue. The example shows that a linear fit can be only a starting point as it does not fit the data well. The wavelengths from the linear fit is also given in the table of the pixel positions at the end of each line. If this number is off completely, then probably a line was misidentified by the user.
Once the wavelength for a few lines in a few orders has been given, the wavelength solution can be calculated. The offset between this order and the real order from the grating equation has to provided (the exact number is not needed). The results from the wavelength solution are used to plot the lines of the emission line catalogue in all three orders shown in the plot in red (lines used for the fit) and green (lines available (only the brightest 30 lines)). For each pixel position the wavelength is calculated and given after the line properties. If only few orders are given, then low polynomial orders should be used.
If some of the identified lines should be deleted, then the user can mark the box at the front of the list entries and then update the plot. At the bottom of the list is the possibility to give pixel and wavelength for non-identified lines. After each update of the plot up to 3 entries can be added like this. The GUI is saving the data each time the plot is updated and at when accepting the wavelength solution.

4.8 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[7] 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[8] 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 6 to 10.

- Each parameter is explained in detail in the configuration file.

- The input and output parameters of the individual procedures are explained in the file `procedures.py`.

4.9 Detailed 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 (→ Figure 6). 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 offset between each order in the trace1 and trace2 file (→ Figure 7).

4. Create the wavelength solution for the night on the calibration fiber spectra (and maybe the science fiber spectra). (→ Figure 8 and 10). This is based on the solution of the previous data set, which is adjusted according to the given parameters (see 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 at different times during the night.)

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

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 offset between this frame and the traces (e.g. which offsets allows the extraction of maximum flux).

   c) Extraction of the apertures for the science fiber.

   d) (for bifurcated fiber spectrographs) Extraction of the apertures for the calibration fiber.

   e1) Finding the offset between the emission lines in the calibration spectra and the lines used for the wavelength solution (for bifurcated fiber spectrographs).

   e1) Additionally, if possible the measured drift from step 6 is interpolated to the mid exposure time and applied. Calculating the wavelength for each pixel in the extracted spectrum.

   f) Creating the blaze 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*).

   k) Write subsets of data into different files to create compatibility with other software.

8. (Optional) Perform the radial velocity analysis.

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 8.)

4.10 Further settings for the parameters

The following steps give an overview about additional parameters which are not covered by 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 (see Fig. 5). 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[9] `%Y-%m-%dT%H:%M:%S`. More formats can be added in the procedure *get_obsdate*.

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

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

4.11 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.

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

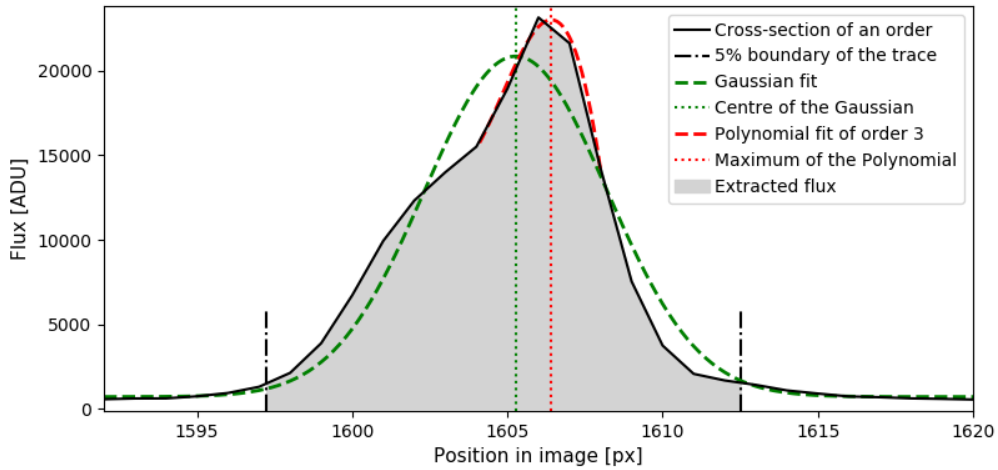

Figure 5: A cross section through one order in a frame to determine the traces of the orders (solid black). The centre of the order is determinded from the highest flux in a polynomial fit (red dotted vertical lines and red dashed line). The extraction boundarys (black dotted-dashed vertical lines) are set to where the flux reaches 5% of the flux range (**width_percentile = 5**), using the minimum flux as zero point.

- 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 offset 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 4.13). 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.
- (optional) Emission line spectra all fibers (not necessarily at the same time) to measure the drift spectrum along the detector.

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).

4.12 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 4.13).
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.

4.13 Creating a list of objects

This is done in the procedure `file_assignment.py`. However, the user can also provide their own list 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_paths` (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 comma-separated values need to be given:

1. Object name (the filename should start with the object name in order to be used)
2. RA (hh:mm:ss.ss or 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. Enable/Disable flag (1 for enabled)
7. Mask to use for the CERES radial velocities. At the moment G2, K5, or M2 are possible options. If empty G2 is used.
8. Velocity width to broaden the lines of the binary mask in CERES. Can be empty.
9. Epoch of the coordinates (only the number, e.g. 2000 or 2015.5)

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.

4.14 Format of the extracted files

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

4.14.1 raw-data filename in path_extraction

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 in barycentric system.
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.
9. Wavelength for each order and pixel without barycentric correction.

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

4.14.2 path_harpsformat + Object name

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.

4.14.3 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.

4.14.4 raw-data filename + _lin_cont

This file contains data in the same form as described in Chapter 4.14.3, only that the continuum corrected data was linearised.

4.14.5 path_extraction_single

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.

4.14.6 path_rv_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-lYghFwpwtdXG9b4I` `view`). The barycentric corrected wavelength solution and the continuum corrected spectrum (multiplied by the bad-pixel mask) are used. See 4.16.1 for more information.

4.15 Results from the pipeline

4.15.1 Preparing the data for one night

For each of the steps given in Chapter 4.9 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 (first searched by Gaussian fit and then more precise by a the maximum of a third order polynomial around the maximum) 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.
- 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 Figure 6.

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: *arctraces_in_master_traces_cal.png*). An example is show in Figure 7.

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 average of t
Info: A 2d polynom fit with 5 orders in dispersion direction (along the traces) and 5 orders in cross-dispersion direction was us

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
```

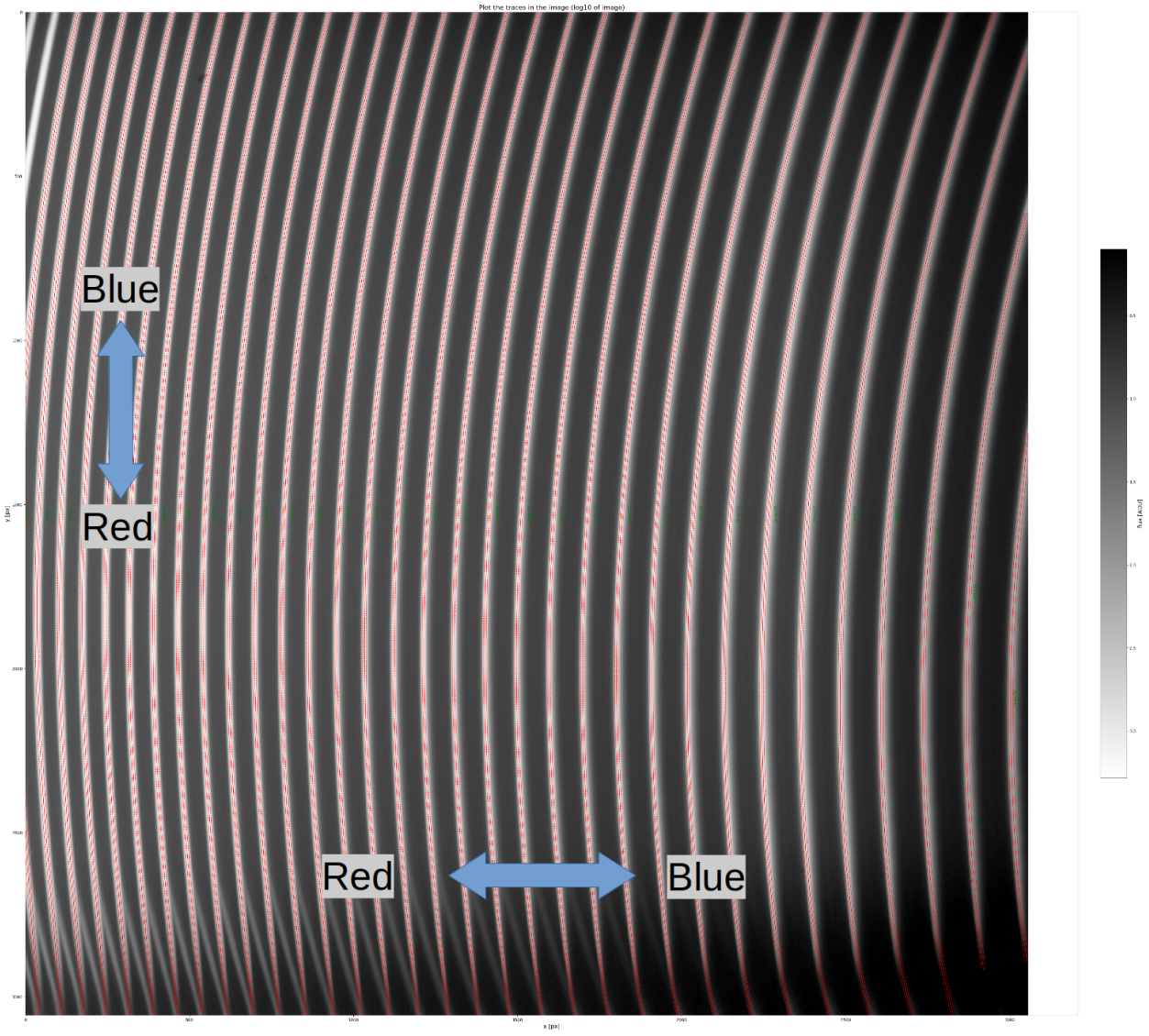

Figure 6: Reduced CCD image of a white light flat (log10 gray scale) with the marked traces of the identified scientific orders/apertures (red). The extraction width (determined from the width given in parameter **width_percentile** of the order multiplied with the value in parameter **extraction_width_multiplier**) is given in the dashed lines.

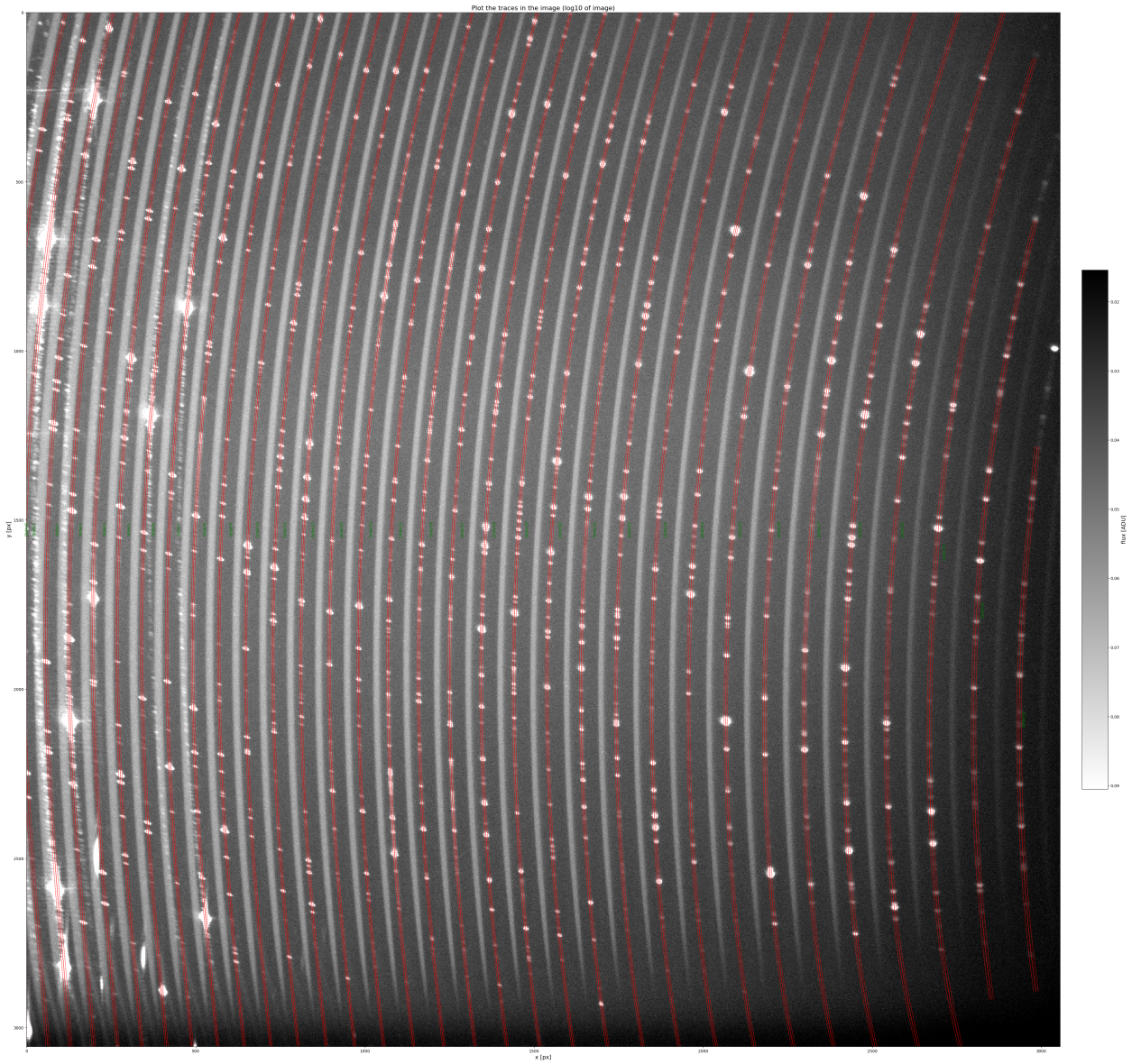

Figure 7: Reduced CCD image of a ThAr exposure (log10 gray scale) with the marked traces of the identified calibration orders/apertures (red). The extraction width (determined from the width given in parameter **width_percentile** of the order multiplied with the value in parameter **extraction_width_multiplier**) is given in the dashed lines.

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

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 4.14.1

4.16 Radial velocity analysis

If the packages are installed, the radial velocity analysis will be run after the extraction of the science files. The necessary files will created each time the `hiflex.py` is run. It is possible to add files from different nights together

4.16.1 TERRA

All information is relative to the folder given in parameter `path_rv_terra` (standard: terra_rv). The data of the individual objects are stored 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 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 (1) rerunning `python <path to scripts>/hiflex.py` or (2) alternatively run the commands as stored in the logfile. Please note that in order to this, the number of orders and the number of extracted pixel has to be always the same.

It is also possible to include the information from other nights by linking or copying the files from folder given in parameter `path_extraction` (standard: extracted) into the current one. Afterwards run `python <path to scripts>/hiflex.py` again. Please note that in order to this, the number of orders and the number of extracted pixel has to be always the same.

It is also possible to rerun TERRA with less images (e.g. exclude bad signal to noise files). To do this change into the folder given in parameter `path_rv_terra`. Remove the files you don't need from the subfolder *object name + /data/*. Afterwards run TERRA using the command given in the logfile. Afterwards move back to the folder in which HiFLEx was run and run `python <path to scripts>/measurements_to_output.py`.

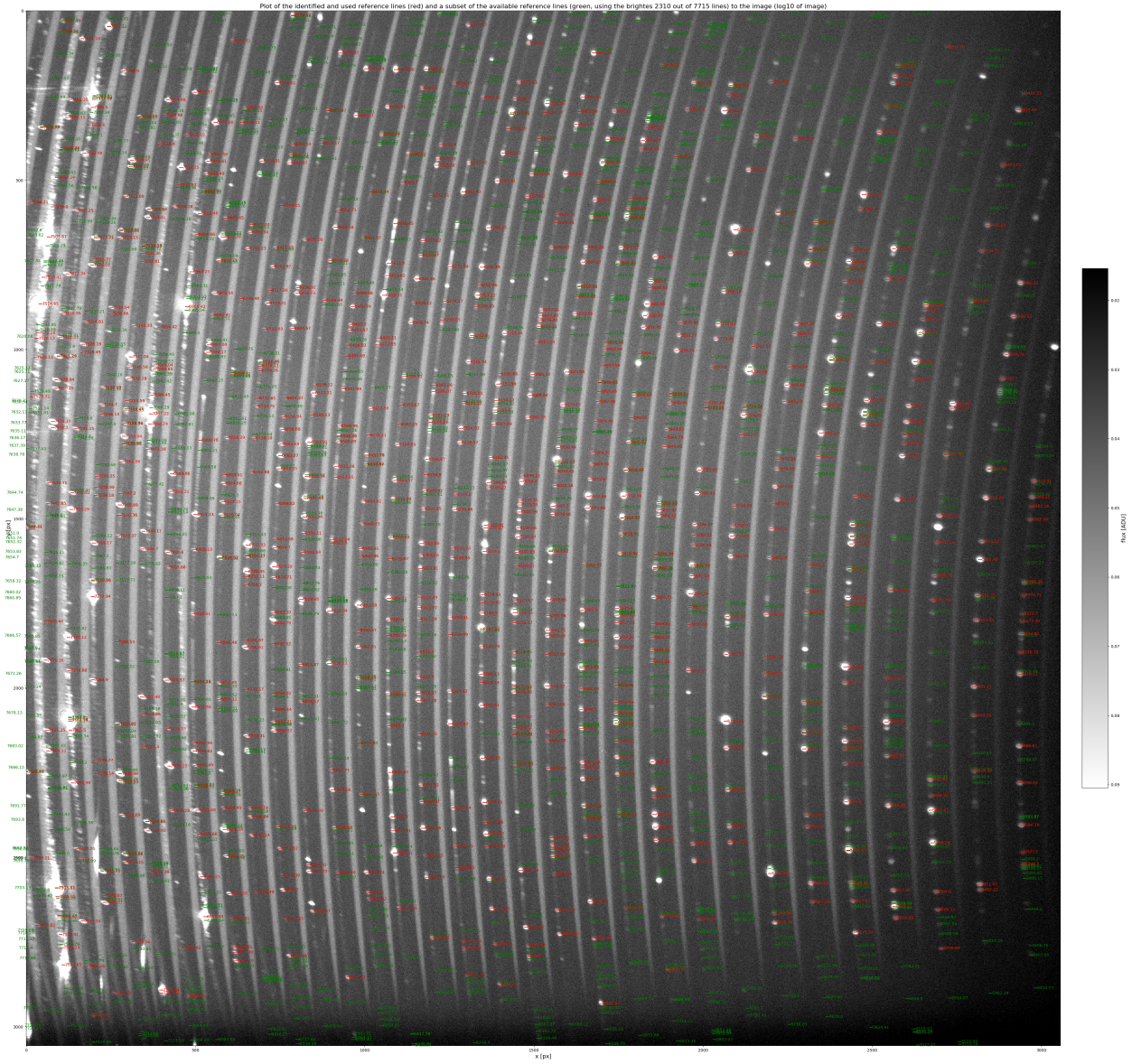

Figure 8: Reduced CCD image of a ThAr exposure (log10 gray scale) with the identified lines from the reference catalogue (red). Only this set of data was used to create the wavelength solution. The remaining lines of the reference catalogue, which weren't used for fitting the solution are shown in green. The data on an order by order basis is shown in Figure 9.

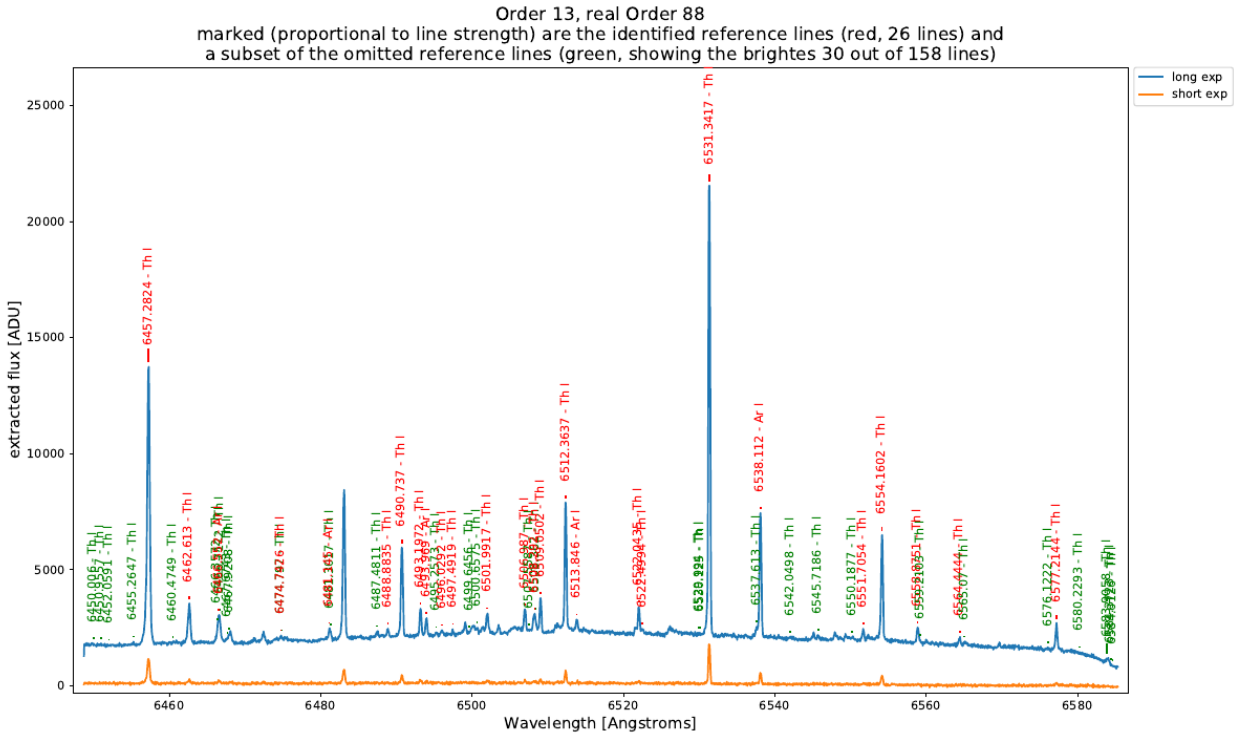

Figure 9: One page (aperture 13, real order 88) of the spectral atlas (file created from parameter **logging_arc_line_identification_spectrum**) created from the wavelength solution. Blue shows the spectrum of the emission lines in the long and orange the emission lines in the short exposure. The identified 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.

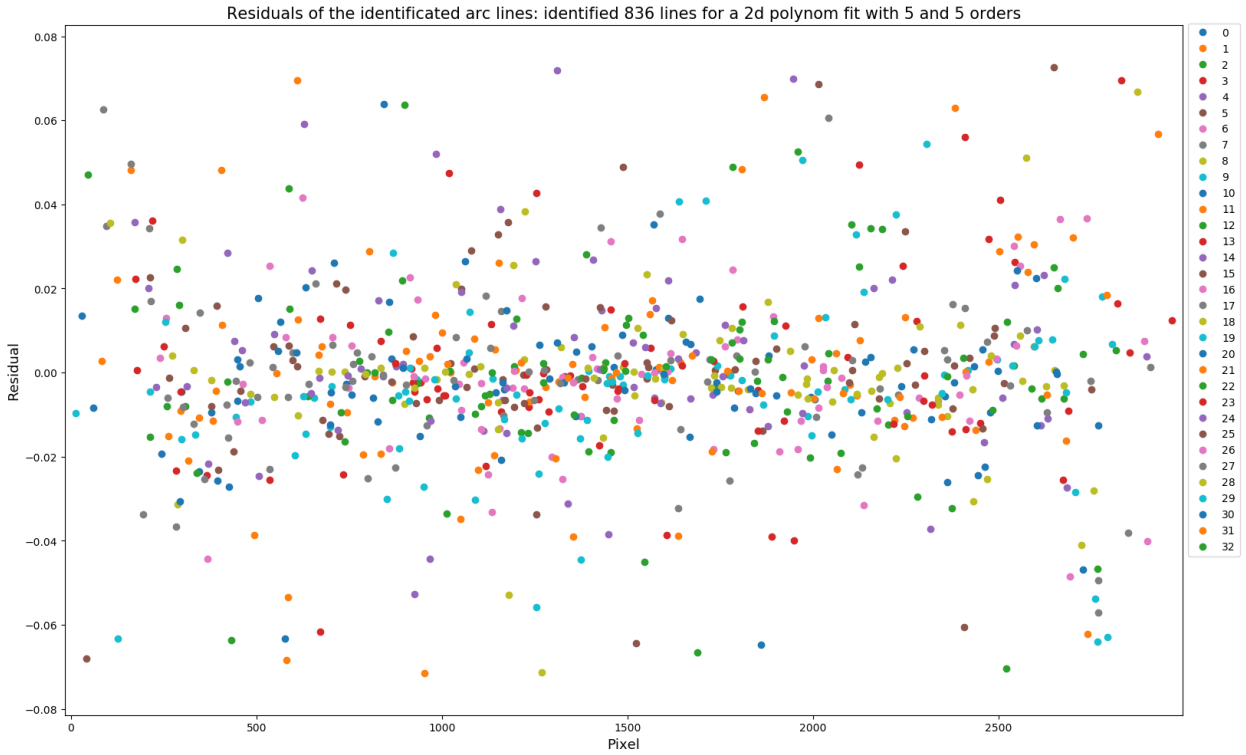

Figure 10: Residuals between the identified catalogue lines and the wavelength solution. Colorcoded are the different apertures.

4.16.2 SERVAL

All information is relative to the folder given in parameter `path_rv_serval` (standard: serval_rv). The data to be used for each Object is stored in the files *filelist_ + object name + .txt*. If SERVAL is installed and `path_serval` is set correctly, then SERVAL will run on each *object name* within the folder. The results of the radial velocity analysis will be stored in *object name + / + object name + rvc.dat*.

It is possible to include the information from previous nights of the same object by linking or copying the extraced files into the corresponding *data/*-folders and (1) rerunning `python <path to scripts>/hiflex.py` or (2) alternatively run the commands as stored in the logfile. Please note that in order to this, the number of orders and the number of extracted pixel has to be always the same.

It is also possible to include the information from other nights by linking or copying the files from folder given in parameter `path_extraction` (standard: extracted) into the current one. Afterwards run `python <path to scripts>/hiflex.py` again. Please note that in order to this, the number of orders and the number of extracted pixel has to be always the same.

It is also possible to rerun SERVAL with less images (e.g. exclude bad signal to noise files). To do this change into the folder given in parameter `path_rv_serval`. Remove the files you don't need from the file *filelist_ + object name + .txt*. Afterwards run SERVAL using the command given in the logfile. Afterwards move back to the folder in which HiFLEx was run and run `python <path to scripts>/measurements_to_output.py`.

4.16.3 CERES

The results are stored in the folder given in parameter path_rv_ceres (standard: ceres_rv). 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 within.

***_<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, $v \sin i$, $vel0$

4.17 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 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 `file_assignment.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` (no spaces around the =)).

5. Some parameters can be overwritten during the run time of the script by user input if the GUI is enabled.

5 Technical details to some steps

5.1 Measuring the drift of the wavelength solution

The wavelength solution is only created once, assuming that the user provides a high quality set of spectra for this step. To calibrate the spectrum during the observation the drift can be measured using emission lamp spectra. For each line that was used to create the wavelength solution a Gaussian fit is applied to the expected area in the emssion line spectrum and the pixel-offset measured. The distribution of the offsets is then binned with few different binning widths and a Gaussian fitted. The centre of the Gaussian and the width are used for the corrections.

Depending on the spectrograph and available data, they can be used to

- Measure the offset between science and calibration fiber (at different times during the night).
- Monitor the drift during the night, so the drift can be interpolated for the science spectra (single fiber spectrograph).
- Measure the the drift for the current science frame (bifurcated spectrograph).

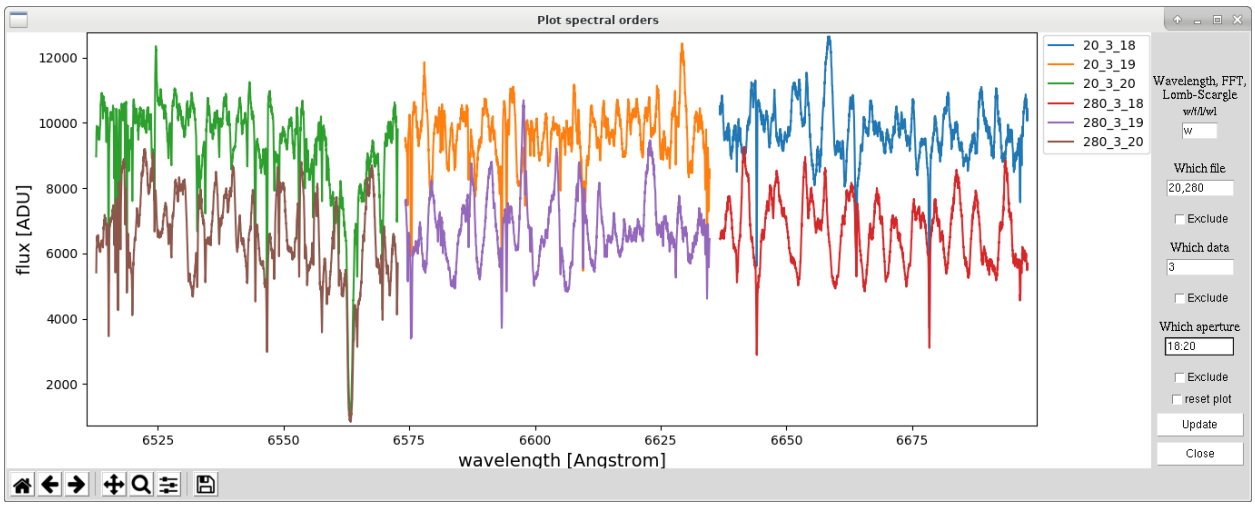

Figure 11: An example of how to plot the extracted data. Shown are the solar spectra of entry 20 and 280 in the plot_files.lst file (*Which file*). The graph shows the blaze corrected flux (*Which data*: 3), plotted over the wavelength (w). Only apertures 18, 19, and 20 are shown. Each of the selections 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 4.14.1 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[10] in the lower left corner. Please refer to Footnote 10 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 2 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 Figure 11 and 12).

Table 2: 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.
w9	(w followed by number): The plot is done against the wavelength (stored in data type 9/number).

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

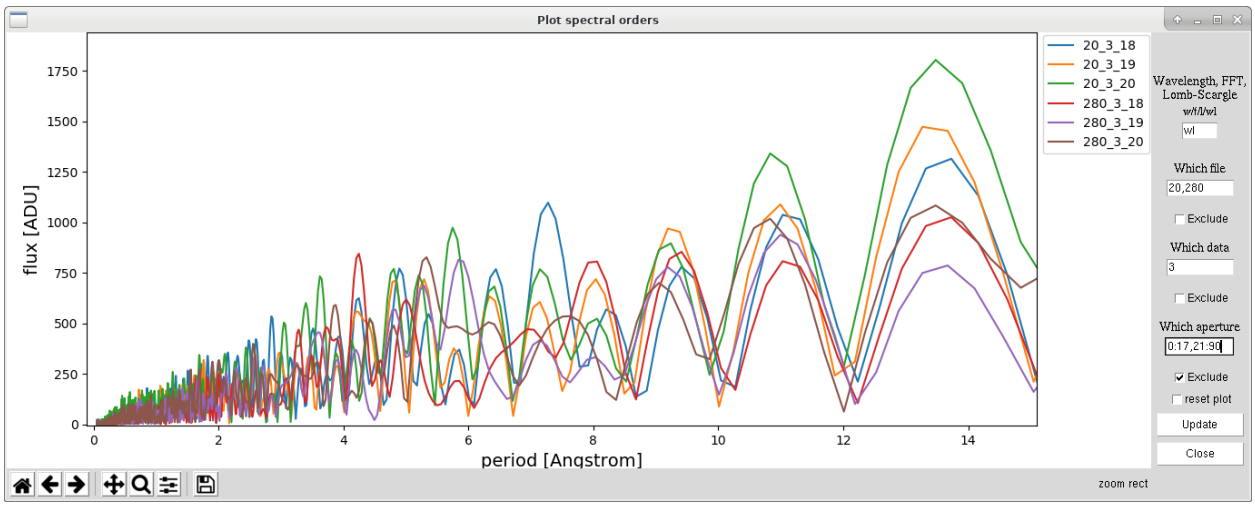

Figure 12: Another example of how to plot the extracted data. Shown is the Lomb-Scargle Periodogram (wl) for the same spectra as shown in Figure 11 of files 20 and 280 in the plot_files.lst file. The blaze corrected spectra (data type 3) of all apertures except 0 to 17 and 21 to 280 (this means only apertures 18 to 20) and their corresponding wavelength solution was used as basis for the calculations.

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, use '2004 07 12' and Program ID '072.C-0488(E)' for calibration data taken by the RV exoplanet search team, 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.
- use_catalog_lines = ThI, ArI, ?, NoID, ArII, ThII
- raw_data_imtyp_keyword = OBJECT
- raw_data_imtyp_bias = BIAS,BIAS
- raw_data_imtyp_trace1 = LAMP,DARK,TUN
- raw_data_imtyp_trace2 = WAVE,WAVE,THAR2
- raw_data_imtyp_blaze = LAMP,LAMP,TUN
- 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, `file_assignment.py` and `hiflex.py` can be run as described above.

7.1.3 HARPS data reduced by the CERES pipeline

The results 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 HiFLEx 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 HiFLEx 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[11]:

- 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

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

The following files are available:

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

HARPS.*_bis_<sptype>_A.fits Bisector from cross correlation with the <sptype> mask.

HARPS.*_ccf_<sptype>_A.fits Cross correlation function matrix for mask for <sptype>.

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

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[12]. 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 HiFLEx format by using the script `reduced_harps_to_hiflex.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 a few percent 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.

7.1.5 Analysing the reduced HARPS data with TERRA

For more information about TERRA please refer to the literature[13]. 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.

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_imtyp_blaze = NA`

[12] See page 22 in https://www.eso.org/sci/facilities/lasilla/instruments/harps/doc/DRS.pdf
[13] https://ui.adsabs.harvard.edu/#abs/2012ApJS..200...15A/abstract

- 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

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 `hiflex.py` can be run again.

```
rm master_*
rm logging/sci_trace1_binned.fits
rm master_traces_sci.fits
rm master_traces_cal.fits
rm logging/arc_lines_found.txt
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 wavelength 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 polynomial 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 (wavelengths in red colour, marker lengths is an indication for the catalogue 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_wavelenso` 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[14] with a description of the problem and the error message. Access to the reduction folder would be preferable.

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

9 Installation and dependencies

This program was written to work with modules installed in Anaconda and Python (3.8 or 2.7). It is recommended to use an installation of anaconda to use this program (If this is not possible, the whole module dependency is given in Appendix B).

9.1 With Anaconda

Anaconda can be downloaded from https://www.anaconda.com/products/individual. After download it can be installed by running

```
sh Downloads/Anaconda3-2020.07-Linux-x86_64.sh
```

(no superuser permissions are needed, the version might need to be changed). Anaconda will ask if *Do you wish the installer to initialize Anaconda3 by running conda init?*, which is recomended to answer with yes, otherwise you might need to follow the steps below

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. Check with `echo $0` what environment you are using.

bash : add the following line (replace */home/hiflex* by the path you used for the installation) to your `.bashrc`.

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

tcsh : add the following line (replace */home/hiflex* by the path you used for the installation) to your `.tcshrc`

```
setenv PATH /home/hiflex/anaconda3/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 Creating an environment for hiflex

Once Anaconda is installed a new environment with the required packages can be created:

```
conda create --name hiflex python=3 numpy scipy matplotlib astropy pycurl \
      ephem rpy2 tqdm psutil statsmodels
conda activate hiflex
pip install gatspy barycorrpy deepCR PyAstronomy
```

This environment has to be loaded in each terminal in which HiFLEx should be run:

```
conda activate hiflex
```

Creating an environment for Python 2 routines: Some external packages (SERVAL and CERES) only work under Python 2.7. Therefore, if you want to use these packages, it is necessary to create a python 2 environment, called (hiflex_p2):

```
conda create --name hiflex_p2 python=2.7 numpy scipy matplotlib astropy pycurl \
      ephem rpy2 tqdm psutil statsmodels
conda activate hiflex_p2
pip install gatspy barycorrpy==0.2.2.1 PyAstronomy
conda deactivate
```

9.2 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.

9.3 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/HiFLEx/releases):

```
wget https://github.com/ronnyerrmann/HiFLEx/archive/v1.2.0.tar.gz
tar -xzvf v*.tar.gz
```

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

file_assignment.py Script to automatically assign the fits files into the corresponding calibration steps.

hiflex.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 TERRA

If you make use of the Radial Velocity analysis from the TERRA , please cite Anglada-Escudé et al. 2012[15].

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 <full/path/to/terra>/terra/PRV.jar
```

This should produce the entry **** TERRA v1.8 **** before failing. If not, please check that java is installed

```
java --version
```

9.4.2 SERVAL

If you make use of the Radial Velocity analysis from the SERVAL , please cite Zechmeister et al. 2018[16].

To install SERVAL please follow the instructions as given at 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, hence for testing you might want to load the Python 2 environment.

SERVAL can be controlled with an instrument file. This has to be provided in the serval `src` folder. Below is a way to create a soft link it when following the installing instructions.

```
ln -s <full/path/to/hiflex>/inst_HIFLEX*.py $SERVALHOME/serval/src/
```

Alternatively, the following line can be used:

```
ln -s <full/path/to/hiflex>/inst_HIFLEX*.py <path/to/mzechmeister>/serval/src/
```

If the SERVAL RV analysis fails, try to run it with a subset of orders for the `oset` parameter. The logfile and cmdhistory.txt in the folder given in `path_rv_serval` give the commands that were run automatically by the pipeline.

[15] https://adsabs.harvard.edu/abs/2012ApJS..200...15A
[16] http://adsabs.harvard.edu/abs/2018A&A...609A..12Z

9.4.3 CERES Pipeline

If you make use of the Radial Velocity analysis from the CERES , please cite Brahm et al. 2017[17].

The Package can be downloaded from https://github.com/rabrahm/ceres. To make it work, please follow the Installation Chapter in the README.md. CERES requires python 2.7. Please note that the compilers in the CERES install process ignore any anconda settings and go straight to the system packages (e.g. for SSEPhem), hence you might want to install it when anaconda is not loaded. The following lines worked in Mai 2020:

```
pip install scipy numpy
git clone https://github.com/rabrahm/ceres.git
cd ceres
python install.py
mkdir data/COELHO_MODELS
cd data/COELHO_MODELS
wget http://www.astro.puc.cl/~rbrahm/coelho_05_red4_R40.tar.gz
tar -xf coelho_05_red4_R40.tar.gz && rm coelho_05_red4_R40.tar.gz
```

It might be necessary to install further packages, e.g.

```
sudo apt-get install libgfortran3 libgsl-dev
```

Problems that might occur when installing CERES

To solve `from pip import main  ImportError: cannot import name main`:

```
sudo python -m pip uninstall pip
sudo python -m pip install pip
```

To solve `jplephem.cc:2:10: fatal error: arrayobject.h: No such file or directory` rename in `utils/SSEphem/jplephem.cc` line 2 from `#include "arrayobject.h"` to `#include "numpy/arrayobject.h"` .

Problemfixing CERES

If the script crashes with:

```
ValueError: math domain error
```

The problem is that the variable `c` is negative (scipy.integrate.simps can create negative values[18]. This is a problem if there are big gaps in wavelength). You might want to search for math.sqrt in `ceres/utils/Correlation/correlation.py` and put the line with CCF in an *if clause*, e.g. around line 110 and 160:

```
        if b*c >= 0:
            CCF  = CCF+(a/math.sqrt(b*c))*(Ll[-2]-Ll[1])
            NOR = NOR+Ll[-2]-Ll[1]
```

9.5 With Anaconda and python 2.7 (until v1.1.0)

Note: Python 2.7 support is ending 2020. Anaconda can be downloaded from https://www.anaconda.com/distribution/#linux. 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. Check with `echo $0` what environment you are using.

bash : add the following line (replace */home/hiflex* by the path you used for the installation) to your `.bashrc`.

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

tcsh : add the following line (replace */home/hiflex* by the path you used for the installation) to your `.tcshrc`

[17] https://adsabs.harvard.edu/abs/2017PASP..129c4002B

[18] https://stackoverflow.com/questions/36803745/python-simpsons-rule-negative-answer-for-positive-area-under-the-curve

```
setenv PATH /home/hiflex/anaconda2/bin:$PATH
```

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

9.5.1 Making an environment for hiflex

Once Anaconda is installed a new environment with the required packages can be created:

```
conda create --name hiflex python=2.7 numpy scipy matplotlib astropy pycurl ephem \
      rpy2 tqdm psutil statsmodels
conda activate hiflex
pip install gatspy barycorrpy==0.2.2.1 PyAstronomy multiprocessing
```

This environment has to be loaded in each terminal in which HiFLEx should be run:

```
conda activate hiflex
```

A Changelog (old)

Please found the changes of newer versions in Chapter 3.

v0.4.3

- GUI for creating a new wavelength solution
- Bug-Fixing

v0.4.2

- Change to the new name: HiFLEx
- GUI for preparing the files
- *localbackground* became *background*
- 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.
- Added parameter `raw_data_object_name_keys` to conf.txt
- 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.

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.

v0.3.4

- 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.

v0.3.3

- Added parameter `raw_data_timezone_cor` to conf.txt.

B Module dependencies if not using Anaconda

This program relies on the following modules in python 2.7

- os, sys, time, datetime
- operator, copy, random, warnings
- tqdm
- pickle, json, urllib2
- numpy, astropy, scipy, ephem
- matplotlib
- Tkinter
- collections
- math
- statsmodels
- psutil, glob
- barycorrpy

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.