European Organisation for Astronomical Research in the Southern Hemisphere

Programme: Directorate Of Engineering (DOE)

Project/WP: Science Operation Software Department

CRIRES+ Pipeline User Manual

Document Number: ESO-413896

Document Version: 1.6.12

Document Type: Manual (MAN)

Released on: 2026-03-26

Document Classification: Public

Prepared by: Jung, Yves

Validated by: Castro, Sandra

Approved by: Ballester, Pascal

Name

ESO – Karl-Schwarzschild-Str. 2 – 85748 Garching bei München – Germany
www.eso.org

Authors

Name	Affiliation
Thomas Marquart	Uppsala University
Alexis Lavail	Uppsala University
Yves Jung	ESO

Changelog

Version	Date	Affected Section(s)	Remarks
0.9.9	2021-10-01	All	First public version
1.1.0	2022-02-24	4.1.2, 5.3, 5.4, 7.2, 9	New observing modes in QuickStart, complete previously missing information, wavecal, detlin, data files table
1.1.4	2022-03-21	7.2.5, 7	detlin noise, sections change numbering
1.2.4	2023-02-15	4.1.5, 6.1, 6.2, 8.2.2	IDP, superresolution, slit-shape
1.3.0	2023-04-25	9, 6.1	Troubleshooting, remove `OPT_VERT`
1.4.0	2023-11-15	4.1.5, 5.3.5, 6.1.3, 7, 8, App D	IDP format, Polarimetry diverging beams, Errors, FAQ, UNE line selection
1.4.2	2023-12-18	5.4.1, 8.3	BPM refinement, Barycentric correction
1.6.2	2024-10-16	5.4.4	Blaze correction
1.6.7	2025-03-12	6.1.3	Note on error calculations

Contents

1 Introduction

1.1 Scope

This document is the user manual used to process of CRIRES**+** observations with the CR2RES Pipeline Recipes, also known as the CRIRES**+** Data Reduction Software (DRS).

The examples on running individual pipeline recipes in this manual use the *EsoRex* command and manually created list of input files. Several interfaces to automatically organise the data, create the list of input files and execute the pipeline recipes in the proper sequence are available, see the ESO Pipeline web page *https://www.eso.org/pipelines* for details.

Please note that the use of Gasgano as a GUI for processing data is deprecated. Its use is no longer recommended and the related section in this manual, as well as support for Gasgano as a data processing GUI application in general will be dropped entirely in a future release.

1.2 Acknowledgements

We are thankful for comments on this manual from Jonathan Smoker and Ulf Seemann.

Special thanks also to Nikolai Piskunov and Ansgar Wehrhahn for their extensive contributions of code to the DRS.

1.3 Stylistic conventions

Throughout this document the following stylistic conventions are used:

bold	in text sections for commands and other user input which has to be typed as shown
italics	in the text and example sections for parts of the user input which have to be replaced with real contents
`teletype`	in the text for FITS keywords, program names, file paths, and terminal output, and as the general style for examples, commands, code, etc

In example sections expected user input is indicated by a leading shell prompt.

In the text **bold** and *italics* may also be used to highlight words.

1.4 Notational Conventions

The DRS for the original, pre-upgrade CRIRES (oCRIRES) used the prefix `crires_` for recipe names ([3]). In order to avoid confusion and because the pipeline for CRIRES**+** was rewritten from scratch and cannot reduce data from oCRIRES, a new prefix was chosen: `cr2res_`.

Hierarchical FITS keyword names, appearing in the document, are given using the dot–notation to improve readability. This means, that the prefix “HIERARCH ESO” is left out, and the spaces separating the keyword name constituents in the actual FITS header are replaced by a single dot.

1.5 Reference Documents

[RD01]	ESO-254264 [8]	CRIRES+ User Manual
[RD02]	2021A&A...646A..32P [6]	Optimal extraction of echelle spectra
[RD03]	VLT-MAN-ESO-14200-4032 [3]	oCRIRES Data Reduction Cookbook
[RD04]	1997MNRAS.291..658D [2]	Spectropolarimetric observations of active stars
[RD05]	2019A&A...624A.122C [1]	New wavelength calibration for echelle spectrographs using Fabry-Pérot etalons

2 Definitions, Acronyms and Abbreviations

BPM	Bad Pixel Map
CalibDB	Calibration Database
CPL	Common Pipeline Library
CCD	Charge Coupled Device
DFS	Data Flow System
DRS	Data Reduction System
ESO	European Southern Observatory
EsoRex	ESO Recipe Execution Tool
FITS	Flexible Image Transport System
FOV	Field Of View
FPET	Fabry Pérot Etalon
GUI	Graphical User Interface
HDRL	High-level Data Reduction Library
LSF	Line Spread Function
OB	Observation Block
pixel	picture element (of a raster image)
PSF	Point Spread Function
QC	Quality Control
SDP	Science Data Product
S/N	Signal-to-noise ratio
SOF	Set Of Files
TBD	To be defined
TBC	To be confirmed
TW	Trace-Wave table
UNE	Uranium-Neon lamp
VLT	Very Large Telescope
WCS	World Coordinate System

3 The Instrument

CRIRES**+** is a major upgrade of the original CRyogenic Infra-Red Echelle Spectrograph, henceforth *oCRIRES*. The upgrade foremost introduces

- A cross-diperser unit (CDU) with one grating per band (YJHKLM).
- New detectors, 3x 2048x2048.
- A spectropolarimeter (SPU) with beam-splitters in YJHK for both linear and circular polarization.
- New calibration sources (U-Ne-lamp, Fabry-Pérot, gas cell)

For details about the instrument's function and operation, we refer to the *User Manual* [8] and only give a short overview over the most important changes from the perspective of data reduction.

A look at the echellogram makes the most important aspects apparent:

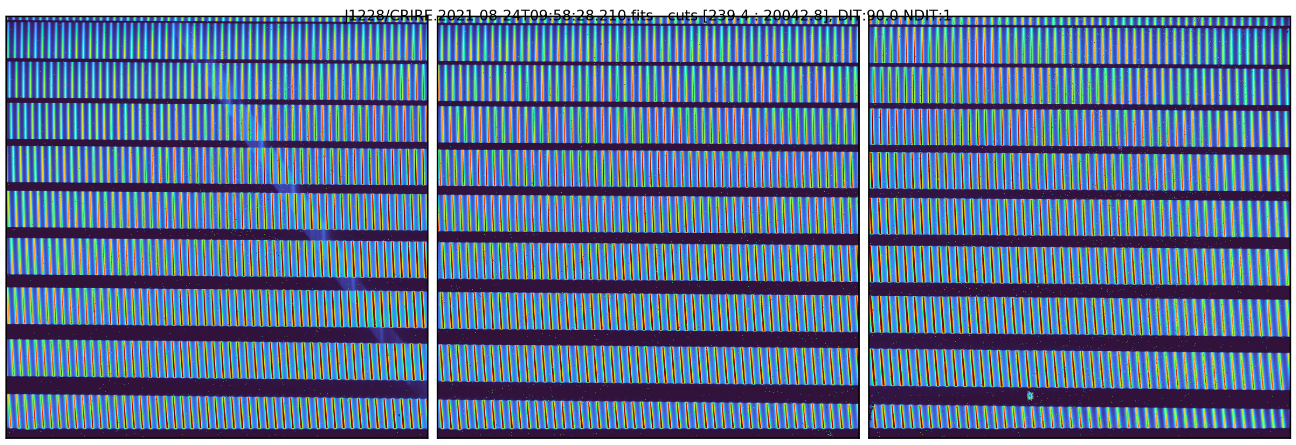

J1228/CRIRE.2021-08-24T09:58:28.210.fits - cuts [239.4 : 20042.8], DIT:90.0 NDIT:1

Figure 3.1: Example raw frame in J-band, with the regularly spaced lines from the Fabry-Pérot etalon.

- There are up to 9 spectral orders (depending on band).
- The orders are spaced unevenly over the detectors.
- The orders are not perfectly horizontal.
- The projection of the slit is tilted, i.e. not aligned with detector columns. And the tilt changes with wavelength.

This in essence made necessary the development of a new DRS from scratch, around robust algorithms for tracing the spectral orders, characterizing the slit tilt and then making use of this information for optimally extracting the spectra into 1D-arrays [6].

Within each order, wavelength increases from left to right over the three detectors. Spectral orders with shorter (longer) wavelengths are located further towards the top (bottom) of the pixel grid, i.e. at higher (lower) Y pixel values.

In order to increase the validity of daytime calibrations for observations from the night, a metrology system is in place that iterates on the position of the echellogram, both in main dispersion and cross-dispersion direction, thereby improving repeatability.

Therefore, and to minimize the calibration overhead, CRIRES**+** is operated with fixed *standard settings*. These are chosen such that the full wavelength range is covered in each band, as well as the detector gaps. In general, the DRS does not rely on these settings, as long as it receives a set of calibrations that match the data. However, only the standard settings are supported for the time being and the DRS uses header keys like `INS.WLEN.ID`[1] for consistency checks, where appropriate.

Y	2
J	3
H	4
K	4
L	7
M	9

Table 3.1: The number of standard settings per band.

[1] `INS.WLEN.ID` takes the form of a string that starts with a letter to indicate the band (YJHKLM), followed by four digits that represent the central wavelength in nanometers, rounded e.g "H1567".

4 Data Reduction Overview

The CRIRES**+** pipeline, like most other recent ESO instrument pipelines, can be executed in three different ways: with *Gasgano*, *esorex* and *Reflex*. Although these are somewhat different environments, it is always the same pipeline, as far as pipeline recipes are concerned. Gasgano is a FITS-file browser and running the pipeline this way is a "manual" task, typically good only for quick look checks.

Reflex executes the recipes via graphic workflow interface which has advantages like sorting data and finding the necessary calibrations automatically, and keeping track of previous executions. The reflex workflows are described in the Reflex Tutorial while this manual uses esorex in its examples on how to run the data reduction cascade.

The details on recipe algorithms and parameters are contained in this document, and the Reflex workflows reduce the calibrations as described in Ch. 5.2, before executing one of the science recipes (Ch. 4.3).

4.1 Data Formats

4.1.1 Detectors and Extensions

Raw data comes as FITS files that contain only headers in the primary extension, and one image extension with the readout result from each detector. The latter are named like `CHIPn.INT1` where `n` is the detector number (1...3), in order from left to right with increasing wavelength in each spectral order.

Data reduction products are either FITS tables or images, depending on whether the product is an image/map or not. The separation into extensions of data from the three detectors is kept, and most recipes treat them independently. For derived maps or images that have errors, three more extension are present, named like `CHIPnERR.INT1`.

Fig. 4.1 illustrates this. The extensions are usually ordered, however we recommend to not rely on the index when opening files with custom scripts or tools. Instead, the FITS extension *name* should be used.

4.1.2 Data Product Headers

The DRS recipes use the FITS headers of the *first raw file* in the SOF as basis for the data products' headers.

Recipes also add some new headers:

- The header keys specific to the data product are named `PRO.*` and includes classification (`PRO.TYPE`, `PRO.CATG`), the list of input files (`PRO.RECi,RAWj` and `PRO.RECi.CALIBj`), and recipe parameters and their values (`PRO.RECi.PARAMj`); with `i,j` being integers increasing from 1.
- Named `QC.*` are the results of quality control parameters, which are used for instrument monitoring. Note that these often reside in the extension headers, since they are derived for each detector.

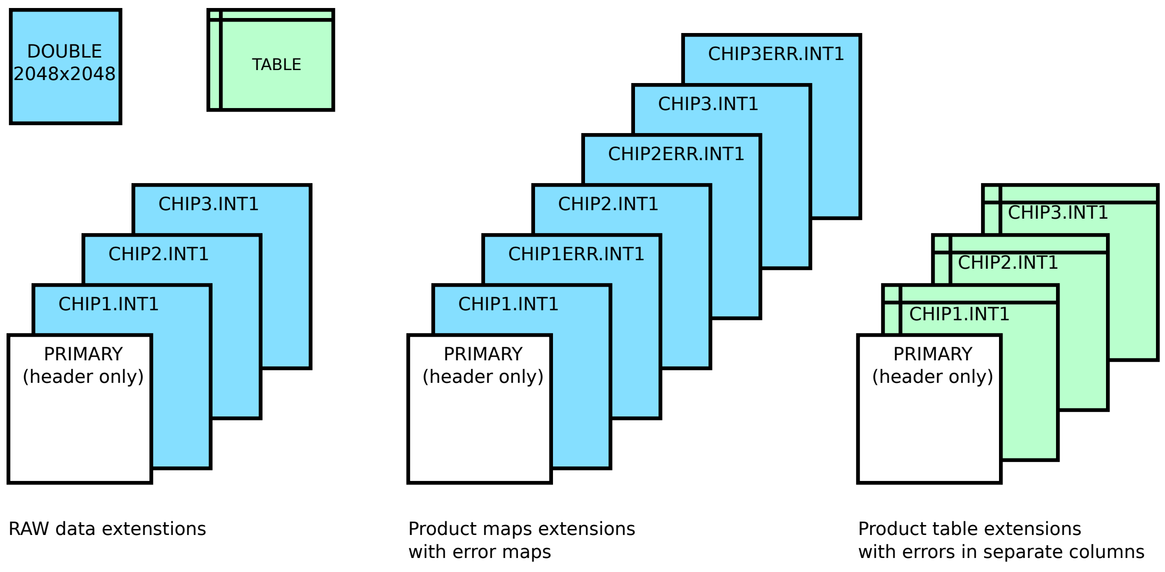


Figure 4.1: FITS extensions in raw frames and data products.

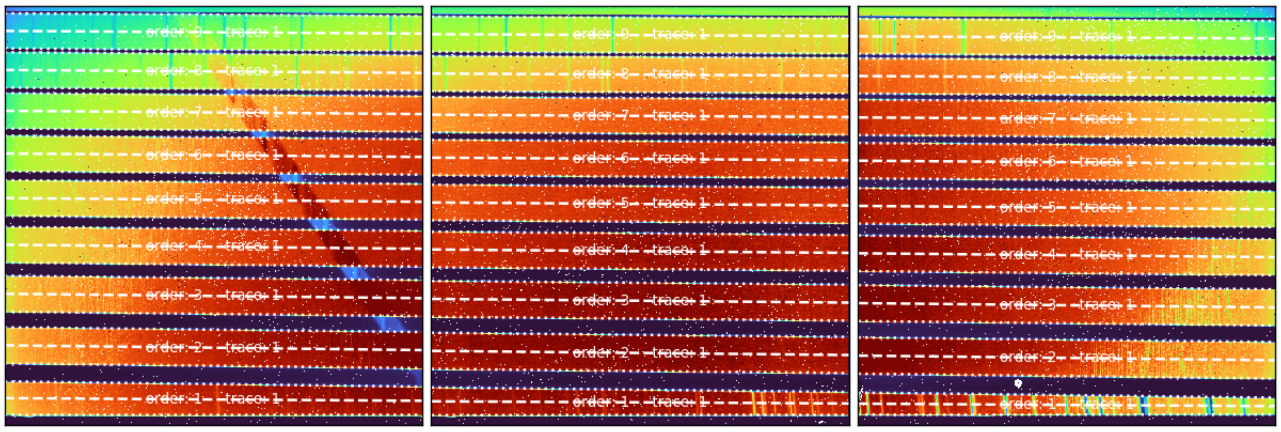


Figure 4.2: An example of a raw flat-field frame in J1228. Overplotted in white is the result of the order tracing, the polynomial fit to the mid-line (dashed) and edges (dotted) for each detector-order.

4.1.3 The TraceWave-Table

The most important FITS table within the CRIRES+ DRS is the *TraceWave*, abbreviated TW henceforth. It contains information about how spectral orders are located in the detectors' pixel grid, how the projection of the slit changes, and the wavelength solution.

Each row in the table corresponds to a single *trace*. Each trace belongs to a single spectral order; an order can have one or more traces, numbered starting with 1 within each order. The *order number* and *trace number* are two of the columns in the TW table, and together they uniquely identify a trace, i.e. a table row.

Each trace stores three polynomials: one for the mid-line, upper and lower edge, respectively. Polynomials are stored as coefficients, lowest power first, and evaluated for pixel columns starting with index 1. Fig. 4.2 shows these polynomials plotted onto a flat-field frame. Associated with each trace is also its *slit-fraction*, that is a number [0.0 ... 1.0] which indicates the height along the slit, ranging from 0.0 at the bottom to 1.0 at the top. Three values are provided, one each for lower edge, mid-line and upper edge of the trace, so that a trace for the full slit has `SlitFraction=[0.0,0.5,1.0]`. Partial slit-fractions are used for nodding and polarimetry; see also Ch. 5.3.2.

Analogous to traces, the wavelength solution is stored as polynomial coefficients that represent the translation from pixel to wavelength space. Because the slit is not vertical with respect to the pixel columns, the wavelength scale naturally shifts within each spectral order. This is taken into account when the spectra are extracted, so that the TW only needs to store the wavelength solution at the mid-line of the trace.

How the orientation and shape of the slit changes along each trace is saved as three "meta-polynomials" that are described in detail in Ch. 6.2.2. Since this is handled automatically by the extraction methods, users normally should not need to get familiar with this aspect.

Several recipes take a TW as input and produce a new one, with some information updated. Notably the wavecal recipe needs a TW as input to know how to extract the spectra of the calibration lamps, and to get a first-guess of the wavelength solution; it will then output a TW with the new wavelengths and the other information propagated from the input.

Naturally, each TW is specific to a single standard setting of the spectrograph. The order tracing and slit tilt are expected to be stable enough to be re-used between nights, especially for observations taken with metrology turned on. The limiting factor is the repeatability of the echellogram (cf. User Manual), but the DRS should not fail to reduce even with significantly mismatched order traces.

4.1.4 Pixels, Indices and Spectral Bins

Pixel index starts at 1, not 0. This is especially relevant when evaluating polynomials from the TW table outside the environment of the DRS.

The spectral bins after extraction correspond to detector columns at the mid-line of the extracted region.

Orders are indexed from 1..9, not always starting at 1, depending on cut-off orders. This index is set by headers in the raw frames that provide first-guesses for the location of spectral orders, and their wavelengths.

To convert the order index to "real" order number *m*, the zero-point that simply needs to be added can be found in `INS.GRAT1.ZP_ORD`. The DRS usually sticks to the index, for example when numbering the spectra in data products. Therefore, unless stated otherwise, order number will refer to the index throughout this document.

4.1.5 Internal data products (IDP)

Since version 1.2.4 (Feb 2023), the CRIRES+ DRS supports the ESO Science Data Products Standard, as per [4].

These additional data products are saved in separate files that carry the prefix `idp_`. By default, the creation of these files is turned off; the flag `--idp` needs to be provided to the recipes `cr2res_obs_nodding`, `cr2res_obs_staring` or `cr2res_obs_pol` to activate this functionality.

`cr2res_obs_2d` does not support IDP.

4.2 The Pipeline Recipes

Our naming convention groups the recipes into

- *Observing recipes*, named like `cr2res_obs_*`, which receive raw science frames and pre-processed calibrations in order to produce the main science data products.
- *Calibration recipes*, named like `cr2res_cal_*`, carry out the main steps necessary to produce the calibration products. They are also triggered online at Paranal.
- The *utility recipes*, named like `cr2res_util_*`, are a more diverse set of recipes. Some offer more fine-grained control over one of the steps within a `cal_` recipe; others exist to prepare static calibrations or perform additional tasks that are not covered by other recipes.

Details about their inputs, outputs and parameters can be found in the *man-pages* (Ch. 10).

In general, calibration and observing recipes will collapse/combine raw frames, when several are supplied. Utility recipes will loop over the given raw frames and produce separate outputs for each of them, unless `--collapse` is set to one of `SUM`, `MEAN` or `MEDIAN`.

4.3 Science reduction

Let's assume for a moment to have reduced calibrations already available and look at the reduction of science frames first.

The `cr2res_obs_*` recipes all have in common that they receive as input a set of raw science frames, and the master calibrations (i.e. output of calibration recipes). They then first apply the calibrations to the raw frames, then combine the ones that belong together, for example the frames from the same nodding position. After that the appropriate algorithms for the data at hand are run to create the output data products.

For the most common use case, nodded observations of point sources, the set-of-files (SOF) to reduce a nodding sequence (three frames each in positions A and B in this example) may look like this:

```
1> cat nodd.sof
$RAW/CRIRE.2021-08-16T23:22:50.334.fits OBS_NODDING_OTHER
$RAW/CRIRE.2021-08-16T23:23:54.728.fits OBS_NODDING_OTHER
$RAW/CRIRE.2021-08-16T23:24:58.960.fits OBS_NODDING_OTHER
$RAW/CRIRE.2021-08-16T23:26:18.394.fits OBS_NODDING_OTHER
$RAW/CRIRE.2021-08-16T23:27:22.616.fits OBS_NODDING_OTHER
$RAW/CRIRE.2021-08-16T23:28:26.956.fits OBS_NODDING_OTHER
$CALIB/K2192_masterflat.fits           CAL_MASTER_FLAT
$CALIB/K2192_tw.fits                   UTIL_WAVE_TW
$CALIB/cr2res_cal_dark_bpm.fits        CAL_DARK_BPM
$CALIB/cr2res_detlin_coeffs.fits       CAL_DETLIN_COEFFS
```

It is then handed to the recipe like this:

```
    1> esorex cr2res_obs_nodding nodd.sof
```

What happens first is that the calibrations are applied, then the frames are combined, A and B separately. Then position B is subtracted from position A, and vice-versa. The background-subtracted spectra get extracted into 1D arrays, using the supplied TW-table for information on order trace, slit-tilt and wavelength calibration.

The other observing recipes accept very similarly looking SOFs, the full list is:

- `cr2res_obs_nodding` for nodding observations of point-sources. Also spectro-astrometry.
- `cr2res_obs_staring` for point-sources without nodding.
- `cr2res_obs_2d` for observations in which the spatial resolution along the slit should be conserved, i.e. no spectrum extraction. This is primarily meant for observations taken with the "generic offset" template.
- `cr2res_obs_pol` for observations with the spectro-polarimeter, including the demodulation into Stokes parameters.

In general, data reduction is performed separately on each detector-order, that is a single spectral order in a single detector. Consequently, results are also stored this way in data products.

4.4 Reducing Calibrations

The following instrument properties need to be characterized:

- Detector: non-linearity (detlin)
- Detector: dark current
- Detector: bad pixel map
- Detector: pixel-to-pixel sensitivity (flat-fielding)
- The location and shape of the spectral orders on each detector.

- The orientation (tilt) of the slit within each order.
- Wavelength calibration.

The *detector non-linearity* is derived from a long series of flat-field frames. These are not taken on a frequent basis and are valid for a long time. Users will almost always be able to use the provided master calibration and do not need to run this step.

The *master dark* is ideally derived from dark exposures with the same DIT as science exposures, and as high NDIT as possible. For short exposures the dark current itself is negligible, but the detector readout-mode leaves a residual pattern that scales non-trivially with DIT, so that scaling dark by unmatched exposure time is not recommended.

DARK exposures are not really "dark", but contain some background thermal emission, especially in the bands redward from K. Therefore, darks are considered specific to each instrument setting.

The *master flat* is derived from a halogen lamp exposure; it is also specific to each spectrograph setting, for obvious reasons. By *flat-field* the DRS means the *normalized flat-field*, that is the pixel-to-pixel sensitivity variations, alone. The wavelength-dependency of the sensitivity within each spectral order is saved as the *blaze* to a separate data product.

Bad pixel masks can come from several steps

- darks, rejecting outliers like hot and dark pixels.
- flats, rejecting pixels outside a range of sensitivity.
- detlin, rejecting pixels with deviating linearity behavior.
- edge pixels. Each detector has 4 columns and rows at the edges that are not sensitive to light.
- inter-order, masking the pixels that are not part of a spectral order.

Therefore, several recipes produce BPMs and there are recipes that can merge and separate them.

Except for the detector linearity, calibrations are valid only for a single *standard setting*. While the DRS should in principle be able to handle non-standard settings just fine, this is not supported for the time being.

Apart from the observing recipes, listed above, there are the following recipes:

- `cr2res_cal_dark` makes master darks and BPMs for each DIT it receives.
- `cr2res_cal_flat` reduces flat-field frames. The normalized flat is its main output.
- `cr2res_cal_wave` calibrates the wavelength scale, using UNE and FPET exposures.
- `cr2res_cal_detlin` derives the detector linearity correction from a series of flat-field exposures in different settings.
- `cr2res_util_calib` applies reduced calibrations to raw science frames. It can also combine frames.

- `cr2res_util_trace` finds the locations of spectral orders and saves them as a TW.
- `cr2res_util_slit_curv` uses an FPET frame to characterize the slit tilt within each spectral order.
- `cr2res_util_extract` extracts 1D-spectra from any input frame along the traces in the given TW.
- `cr2res_util_normflat` makes the normalized flat-field by comparing the frame to the extraction model.
- `cr2res_util_wave` executes a single wavecal method on already extracted lamp spectra.
- `cr2res_util_splice` uses the blaze function to normalize spectra and resamples overlap-regions from different spectral orders or settings into a continuous spectrum.
- `cr2res_util_genstd` converts standard-star spectra from plain-text into FITS tables.
- `cr2res_util_genlines` converts line catalogs from plain-text into FITS tables. Also makes sub-catalogs that only contain lines from pre-selected wavelength regions for each setting.
- `cr2res_util_trace_map` evaluates a TW into maps of the order number and wavelength, and a visualization of the slit-tilt.
- `cr2res_util_bpm_merge` merges BPMs.
- `cr2res_util_bpm_split` splits BPMs into the different types.

4.4.1 The simple way

Fig. 4.3 shows how calibrations can be reduced in a way that matches how the online (quick-look) reductions are run at the telescope. For each type of calibration data there is a single recipe that processes them. Apart from the raw frames, the necessary inputs can either come from the CalibDB or from a previous step.

If a TW is provided to `cr2res_cal_flat`, it will use the information therein; otherwise it performs the tracing of the spectral orders to find their location. However, the slit tilt cannot be determined from flat-field frames which means that the extraction of the blaze function will be done assuming a vertical slit, yielding sub-optimal results in the bands Y-K. Therefore providing a TW is recommended in these bands, either the one from the CalibDB, or one made via `cr2res_util_trace` and `cr2res_util_slit_curv`.

Daily calibrations are taken with metrology turned on, and both the static calibrations and the CalibDB at the telescope are derived from data with metrology. This means that this way of reducing calibrations, relying on the pre-existing TW for order locations and slit-tilt, is not expected to have negative impact on the quality of data products, as long as the night-time observations were also taken *with* metrology.

Example SOFs and commands can be found in Ch. 5.2.

4.4.2 The step-by-step way

Fig. 4.4 shows the calibration data flow step-by-step, using the *util* recipes that each only perform a single task. In addition, no pre-existing calibrations are used, and it is in fact the way the set of static calibrations is assembled in the first place.

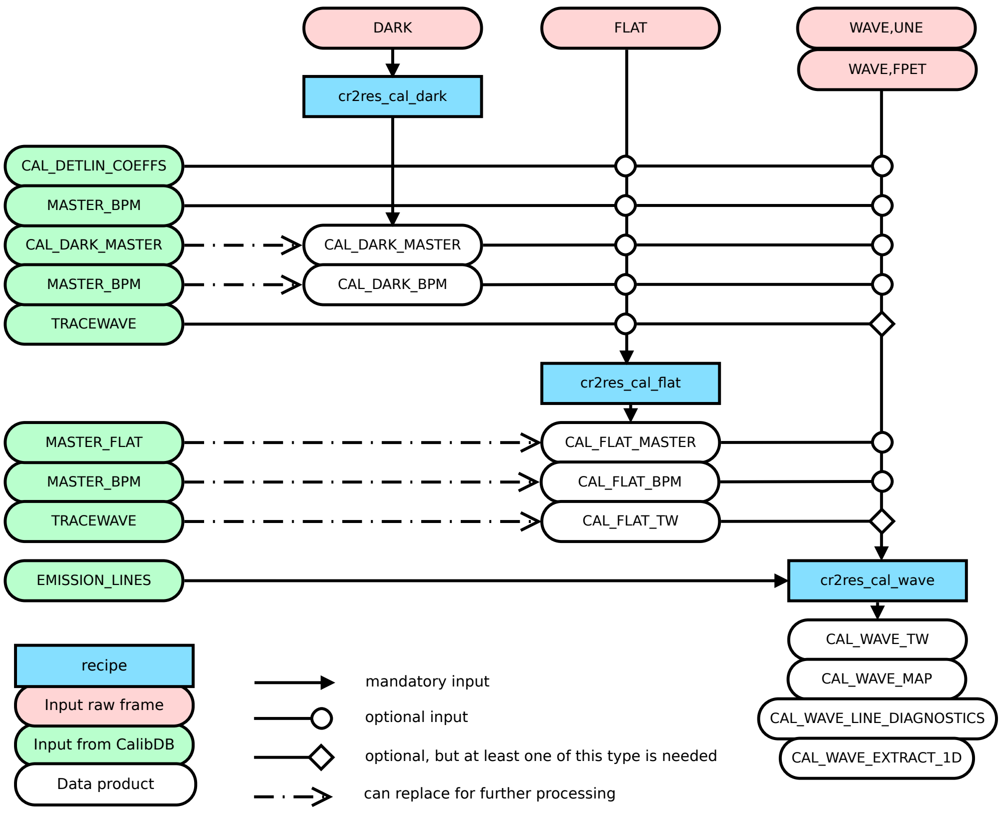


Figure 4.3: Flow diagram for calibration reduction with the *calibration recipes*, assuming a populated CalibDB exists. Arrows and lines indicate the inputs to the recipes (blue), in the form of raw data (red), master calibrations from the CalibDB (green), or intermediate products from previous steps (white). Dash-dotted lines indicate when a CalibDB or intermediate product can be chosen for later steps further to the right. All line connections are optional (open symbols), but generally recommended being present. Also note that open diamonds mark TW inputs, of which at least one needs to be present.

A concrete example, using the same numbering of steps, is shown in Ch. 5.5. Some aspects are worth pointing out first:

1. `cr2res_cal_dark` combines the input DARK frames into a master dark for each setting and combination of DIT and NDIT. Hot and dead pixels get saved as BPM.
2. `cr2res_util_calib` combines (`--collapse=MEAN`) the raw FLAT frames and applies the master dark and BPM.
3. `cr2res_util_trace` finds continuous regions on the detectors that have signal, and fits them with polynomials for the orders' mid-lines and edges. These polynomials we call a *trace* and each order can have more than one trace, e.g. marking different heights along the slit.
4. `cr2res_util_slit_curv` uses the TraceWave-table (TW) from the previous step to measure the

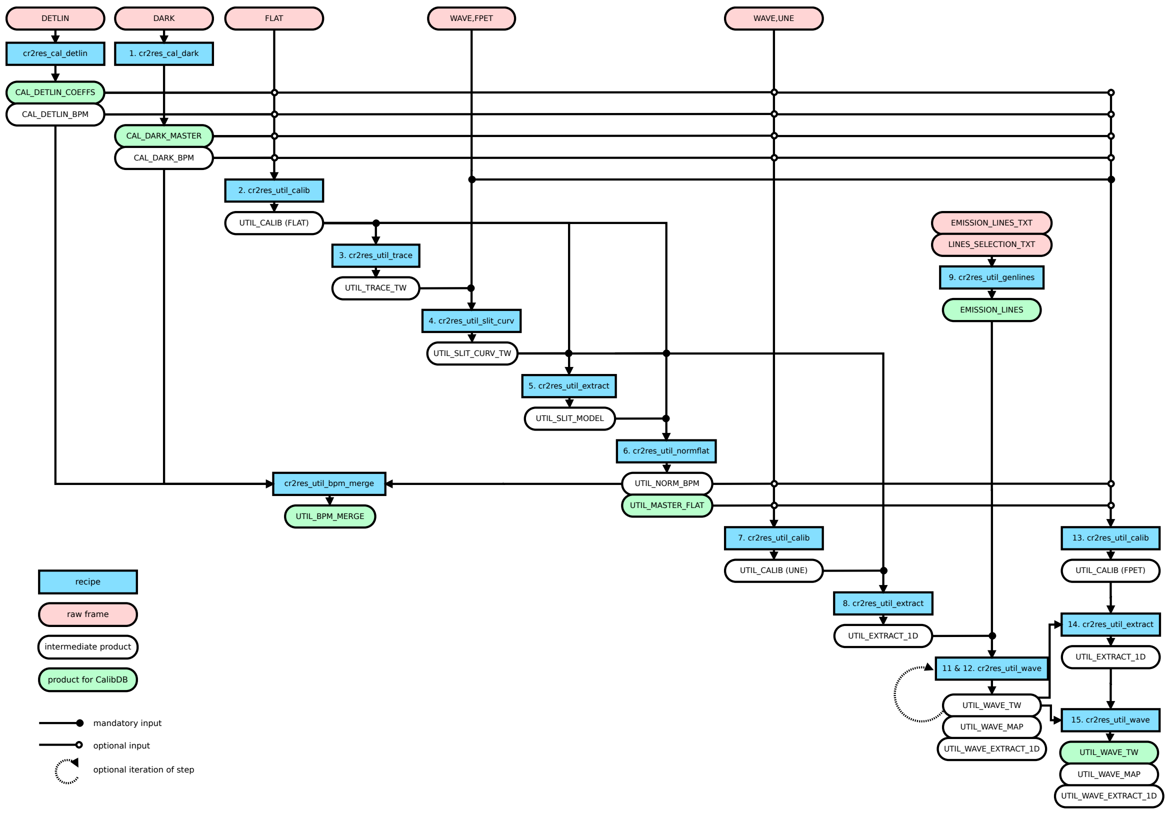


Figure 4.4: Flow diagram for calibration reduction with the *utility recipes*, without relying on static calibrations. In other words, this is how the products for the CalibDB (green) can be made from scratch. Steps are numbered, but execution order can be changed whenever data flow allows. For notes on individual steps see main text.

lines in an FPET frame to determine how the slit tilt changes within each detector-order. The result is stored in an updated TW.

5. `cr2res_util_extract` can now use this TW to extract the calibrated flat-field frame from step 2. This gives us the blaze spectrum (1D) and a 2D model of the frame.

6. This model contains all non-local features, meaning that `cr2res_util_normflat` can calculate the *normalized flat-field* through dividing the model by the original frame. Outliers in sensitivity get flagged as a BPM again.

7. Next we use `cr2res_util_calib` once more, this time to merge the UNE frames and apply the calibrations from previous steps.

8. Now `cr2res_util_extract` can collapse the result from the last step into 1D spectra.

9. `cr2res_util_genlines` creates the catalog files for each setting, based on the base catalog and selection files that flag wavelength regions as usable for the setting at hand.

11. `cr2res_util_wave` uses the spectra and catalog from the last two steps to derive a wavelength solution, i.e. a polynomial that translates pixel-coordinates into wavelength (nm). Available methods include cross-correlation and line-fitting. This step can be repeated, for example to step-wise increase the polynomial degree.

13-15. Analogous to steps 7-11 but for an FPET frame this time. The Etalon method of `cr2res_util_wave` will use the zero-point from the incoming TW and refine the higher degrees of the solution.

5 Data Reduction Cook-Book

5.1 Getting Started with EsoRex

EsoRex is a command-line tool which can be used to execute the recipes of all standard VLT/VLTI instrument pipelines. With *EsoRex* in your path, the general structure of an *EsoRex* command line is

```
1> esorex [esorex options] [recipe [recipe options] [sof [sof]...]]
```

where options appearing before the recipe name are options for *EsoRex* itself, and options given after the recipe name are options which affect the recipe.

All available *EsoRex* options can be listed with the command

```
1> esorex --help
```

and the full list of available parameters of a specific recipe can be obtained with the command

```
1> esorex --help <recipe name>
```

The output of this command shows as parameter values the current setting, i.e. all modifications from a configuration file or the command line are already applied.

The listing of all recipes known to *EsoRex* can be obtained with the command

```
1> esorex --recipes
```

The last arguments of an *EsoRex* command are the so-called *set-of-frames*. A *set-of-frames* is a simple text file which contains a list of input data files for the recipe. Each input file is followed by an unique identifier (frame classification or frame tag), indicating the contents of this file. The input files can be given as absolute or relative path, and *EsoRex* allows the use of environment variables so that a common directory prefix can be abreviated. Individual lines may be commented out by putting the hash character (#) in the first column. An example of a *set-of-frames* is shown in the following:

```
1> cat darks.sof
/data/crires/CRIRES.2019-03-29T09:50:36.645.fits DARK
$RAW_DATA/CRIRES.2019-03-29T09:52:16.513.fits DARK
$RAW_DATA/CRIRES.2019-03-29T09:53:47.996.fits DARK
#$RAW_DATA/CRIRES.2019-03-29T09:55:04.515.fits DARK
#$RAW_DATA/dark5.fits DARK
```

These *set-of-frames* files will have to be created by the user using a text editor, for instance.

Finally, if more than one *set-of-frames* is given on the command-line *EsoRex* concatenates them into a single *set-of-frames*.

5.2 Calibration, made easy

This example follows the data flow described in 4.4.1. It makes use of a pre-existing TW-table, for example the one from the CalibDB, for order traces and slit-tilt. All data needs to match in spectrograph setting.

`cr2res_cal_dark` only receives raw frames, so the SOF looks like the one just above and gets processed like

```
  1> esorex cr2res_cal_dark darks.sof
```

The outputs are master darks and BPM. The recipe parameters influence how outliers are rejected and where to put the threshold for rejecting pixels as bad. The default values should give good results.

Next are the FLATs.

```
1> cat flats.sof
$RAW/flat1.fits          FLAT
$RAW/flat2.fits          FLAT
$RAW/flat3.fits          FLAT
$CALIB/dark_bpm.fits     CAL_DARK_BPM
$CALIB/dark_master.fits  CAL_DARK_MASTER
$CALIB/tw.fits           UTIL_WAVE_TW
cr2res_detlin_coeffs.fits   CAL_DETLIN_COEFFS

2> esorex cr2res_cal_flat flats.sof
```

The main data product is the normalized flat-field (`PRO.CATG=CAL_FLAT_MASTER`). Note that even though the recipe can perform the order-tracting by itself, it is recommended to provide a TW as input; especially in the bands YJHK, where the slit-tilt can be characterized. This way absorption lines affect the flat-field less.

For wavelength-calibration, both the UNE and FPET frames should best be provided at once:

```
1> cat wave.sof
$RAW/une.fits            WAVE_UNE
$RAW/fpet.fits           WAVE_FPET
$CALIB/dark_bpm.fits     CAL_DARK_BPM
$CALIB/dark_2s_master.fits CAL_DARK_MASTER
$CALIB/tw.fits           UTIL_WAVE_TW
$CALIB/cr2res_detlin_coeffs.fits   CAL_DETLIN_COEFFS

2> esorex cr2res_cal_wave wave.sof
```

This produces a new TW, with updated wavelength solutions and retaining information like the order traces from the input TW.

In this simplified reduction, we rely on a pre-existing TW for traces and slit-tilt, and pre-made non-linearity correction. This normally should not negatively impact the quality of the data products, especially when metrology is used to increase the repeatability of the instrument.

5.3 Examples of science reductions

5.3.1 Nodding

An example SOF and esorex-call for `cr2res_obs_nodding` were already shown in Ch. 4.3 and there really is not much more to it. Note that there is no master-dark provided, and indeed it is not recommended doing so, because it likely worsens the results compared to relying on the subtraction of A and B frames from each other for removing detector features and background.

An aspect to consider is how much of an observed nodding sequence is given to the recipe at a time. It will combine all the A-frames and all B-frames (saved as `PRO.CATG=OBS_NODDING_COMBINED[AB]`), then do the pair-wise subtraction and spectrum extraction, resulting in a single set of spectra (from all detector-orders) for A, and one for B (`OBS_NODDING_EXTRACT[AB]`). Therefore, if time-evolution plays a role in the science, a set of data might need to be split up and reduced individually. An equal number of frames for positions A and B always needs to be supplied.

The recipe also produces spectra that are combined from A and B (`OBS_NODDING_EXTRACT_COMB`). This naturally involves resampling and relying on the wavelength scale, because the tilted slit makes the spectral binning different between the top and bottom halves of the slit. This is why the spectra *before* this combination are considered the primary data product. The combined spectrum contains the summed ADUs from A and B.

The jitter around positions A and B does not need to be treated explicitly by the DRS. Instead, it makes use of the fact that the extraction algorithm (cf. 6.1.1) can handle arbitrary slit-illumination functions, in this case the combined frame simply looks like multiple "stars" along the slit.

A few parameters can be tweaked to potentially improve results. If there remain detector features that correlate with the detector columns in the combined frames, as has sometimes been found to be the case, `--subtract_nolight_rows` can be set to `TRUE`. This makes use of the bottom 40 rows of the detectors which are intentionally baffled to not receive light. A vertical median over these rows is calculated, and the result subtracted from the image row-by-row.

The oversampling of the slit-function can be changed via `--extract_oversample`. This affects linearly the calculation effort of the extraction, which in turn is the bulk of the total recipe runtime, so that increasing the oversampling from a factor of 5 to 10 means the reduction takes twice as long. In general, values below 5 give suboptimal results and values above 10 are appropriate when the slit-illumination changes sharply. See Ch. 6.1.1 for more details.

In addition to the aforementioned data products, the recipe also saves two new TW tables with `PRO.CATG=OBS_NODDING_TW[AB]`. These correspond to the lower and upper halves of the slit, i.e. the A and B nodding positions, and thus have their respective slit-fractions set to `[0.0, 0.25, 0.5]` and `[0.0, 0.75, 1.0]`. They can come handy, for example, to re-run the extraction of a `OBS_NODDING_COMBINED[AE` frame, using `cr2res_util_extract`; or to over-plot the traces on the combined frame to check the alignment of the target.

5.3.2 Extracting multiple sources in nodding

In the case when multiple targets are placed on the slit (e.g resolved binary stars) in nodding observations, a few simple steps are needed to extract the spectrum of each target separately. In the remainder of this section, we assume the user wishes to extract the spectra of two stars on the slit. The procedure can be adapted to handle any number of extractions.

Run the `cr2res_obs_nodding` recipe with the default slit height. This step is detailed in Ch. 5.3.1 and an example SOF file is shown in Ch. 4.3. From this step, we can disregard the extracted spectrum (as the recipe extracts a single spectrum for the entire slit). However, we will use the combined pair-wise subtracted A and B frames (`PRO.CATG=OBS_NODDING_COMBINED[AB]`) illustrated in Fig. 5.1.

```
1> esorex cr2res_obs_nodding nodd.sof
```

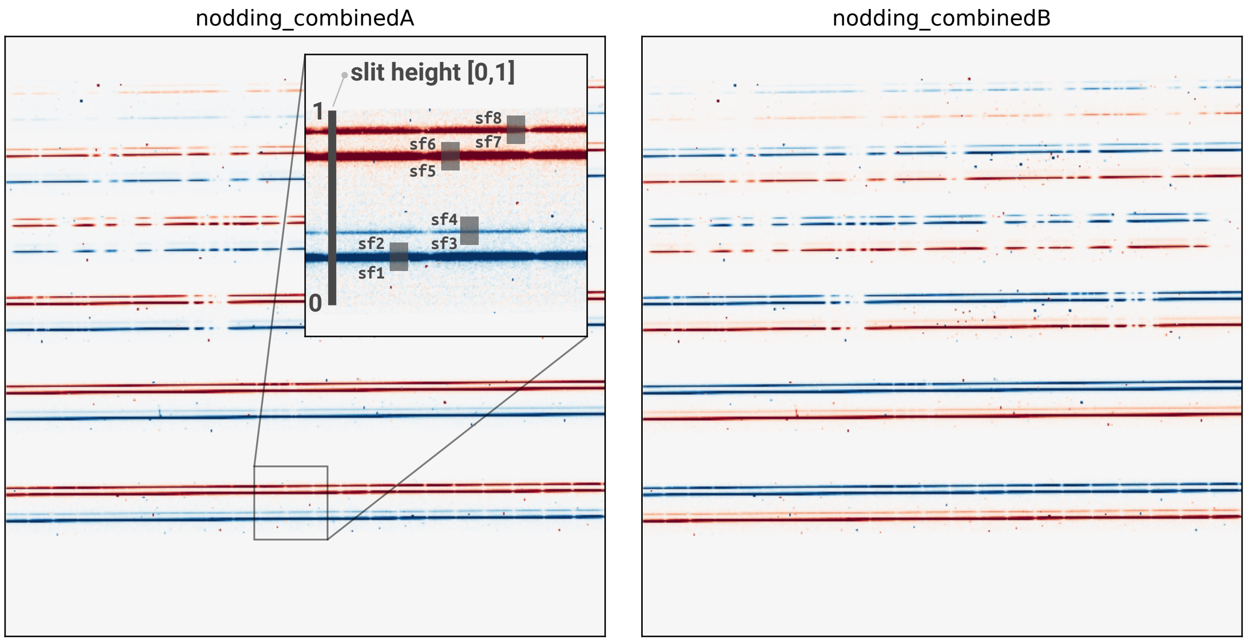


Figure 5.1: The two plots show the combinedA and combinedB frames on one detector. The inset on the left subplot zooms on a part of a spectral order, indicating the full slit height, and illustrating slit fraction values to extract the spectra.

Extract the spectra of the two stars from the nodding position A - hereafter named A1 and A2. This is done by running the `cr2res_util_extract` recipe on the combinedA frame twice, specifying slit fraction values adequate for each of the two spectra, as illustrated in Fig. 5.1.

Repeat the previous step on the combinedB frame to extract the spectra B1 and B2, specifying adequate slit fractions.

```
1> mkdir A1 A2 B1 B1 # create directories to store spectra
2> esorex --output-dir=A1 cr2res_util_extract --slit_frac=0.23,0.30 A.sof
3> esorex --output-dir=A2 cr2res_util_extract --slit_frac=0.35,0.45 A.sof
4> esorex --output-dir=B1 cr2res_util_extract --slit_frac=0.74,0.84 B.sof
5> esorex --output-dir=B2 cr2res_util_extract --slit_frac=0.86,0.97 B.sof
```

The values of the slit-fraction listed here match the example from the figure; they need to be adapted for each observation! The range of `[0.0 ... 1.0]` corresponds to the [bottom ... top] edge

The SOF files `A.sof` and `B.sof` contain only the TW file and

```
    cr2res_obs_nodding_combinedA.fits OBS_NODDING_OTHER
```

for `A.sof`, and the equivalent combinedB frame for `B.sof`. No calibrations needed, because they are already applied to the combined frame.

If the user wants a single spectrum from nodding position A and for each of the star, they still need to combine the A and B spectra – essentially interpolating one to the wavelength scales of the other.

5.3.3 Staring

`cr2res_obs_staring` is very similar to `cr2res_obs_nodding`, except the nodding part. Thus, a master-dark should be supplied in the SOF, in addition to the other files:

```
1> cat stare.sof
$RAW/stare_exp1.fits        OBS_STARING_JITTER
$RAW/stare_exp2.fits        OBS_STARING_JITTER
$RAW/stare_exp3.fits        OBS_STARING_JITTER
$RAW/stare_exp4.fits        OBS_STARING_JITTER
$RAW/stare_exp5.fits        OBS_STARING_JITTER
$CALIB/K2192_masterflat.fits                  CAL_MASTER_FLAT
$CALIB/K2192_tw.fits                          UTIL_WAVE_TW
$CALIB/cr2res_cal_dark_K2192_bpm.fits         CAL_DARK_BPM
$CALIB/cr2res_cal_dark_K2192_30s_master.fits  CAL_DARK_MASTER
$CALIB/cr2res_detlin_coeffs.fits              CAL_DETLIN_COEFFS
```

Ideally, the DARK frames should match the DIT of the science frames.

If the target is always close to the center of the slit, one can decrease the extraction height for speedup.

```
  1> esorex --extract_height=50 cr2res_obs_staring stare.sof
```

Analogous to the example of the binary star above, the option `--slit_frac` can be used to extract only a portion of the slit. This is useful to extract the target and an empty portion of the slit, in order to perform sky-subtraction as part of the analysis. Note that, due to the slit-tilt, the wavelength bins will differ slightly between the spectra, so one needs to be resampled to the other.

5.3.4 Generic Offset

For data taken with the generic-offset-template, the DRS assumes that the spatial resolution along the slit holds valuable information, and therefore does not collapse the data into 1D-spectra. Instead, the recipe `cr2res_obs_2d` produces output tables with 2D spectra, where each in-order and not-bad pixel has an entry with the following values and column names (prefixed with order and trace number as for 1D spectra).

- `POSITIONX`, the original X-coordinate of the pixel

- `POSITIONY`, same for Y
- `SLIT_FRACTION`, from [0..1], the spatial coordinate along the slit, i.e. on sky.
- `WL`, the wavelength
- `SPEC`, the spectrum in ADU
- `ERR`, the error spectrum.

Plotting the against wavelength and slit-fraction as X,Y-coordinates yields a rectified 2D spectrum. Resampling to a regular grid is left to users.

The recipe interface of `cr2res_obs_2d`, and thus the SOF files, follows the same philosophy as the other `cr2res_obs_*` recipes. The same processed calibrations are listed together with the to-be-reduced raw frames.

The sky-pointings are subtracted from the object-pointings like this:

```
if Number of SKY == Number of OBJ
     Loop on OBJ frames
         Subtract the corresponding SKY (same position in the list),
           applying the DIT correction ratio.
         Execute 2D-extraction
else
     Average the SKY frames (if any)
     Loop in OBJ frames
         Subtract the Averaged Sky (if any) regardless of the DIT
         Execute 2D-extraction
```

5.3.5 Polarimetry

For polarimetric observations, the spectropolarimetry unit is inserted into the light path ahead of the slit. It splits the incoming light beam into two parallel beams of orthogonal polarization state, which then pass through a slit mask, also called *deckers*. There are two such deckers: one lets the light through the first and third quarter of the slit (i.e. each hole is 2.5" high), the other lets light through the second and fourth quarter (cf. RD01[8]). These two deckers are designed to cover the zeroth order of the beam splitter in the two nodding positions along the slit.

The recipe `cr2res_obs_pol` is used in a completely analogous way to the nodding-recipe, the SOF looks just like the example from 4.3, except that the raw frames are tagged `OBS_POLARIMETRY_OTHER` instead. In addition, the recipe expects the number of raw frames to be a multiple of 8. This is because the observing template always takes 4 exposures at each nodding position, with alternating rotations of the internal optical elements; and because the number of AB-cycles is expected to be $>$ 0, for background subtraction.

As usual, the provided master calibrations are applied first. For the reduction of position A, the four frames from B are averaged and subtracted, and vice versa. This works because the deckers block the light in the respective opposite quarters of the slit.

Then the two beams are extracted from each frame, each order and detector. This means more extractions than for non-polarimetric observations which is why this recipe takes significantly longer to run.

The order traces for the beam extraction are calculated by taking the wavelength dependence of the beam separation into account, i.e. the traces for the upper and lower beams diverge from each other[2] Separate traces are computed for the beams in nodding position A and B. The default extraction height is set to 32 pixels, less than the decker hole length, because the beams can get close to the edges.

The individual extracted spectra are then combined with each other, following the demodulation scheme described by [2]. The demodulation scheme minimizes instrumental effects and produces the following results, saved in a FITS table that is analogous to non-polarized spectra:

- Intensity spectrum, i.e. the sum of all the beams. Table columns named like i_INTENS where i is the order number.
- The Stokes parameter spectrum, i.e. *V*, *Q* or *U*, normalized by the intensity spectrum. These are stored in the columns i_STOKES. Which of the Stokes parameters was observed in a dataset should be clear from your data organization, but can also be seen from the header INS.POL.TYPE.
- The Null spectrum normalized by the intensity spectrum. This is a diagnostic spectrum from the demodulation that characterizes the level of spurious polarization. In the ideal case, there should be no polarization signatures present in the Null spectrum. i_NULL.
- Error spectra for each of the above, in columns named like these, plus the suffix _ERR.
- The array of wavelengths for each of the spectral bins.

By default, cr2res_obs_pol does not produce intermediate outputs like the calibrated raw frames and the decker-traces for extraction. The recipe parameter --save_group=1 can be used to save these for a single group of 8 raw files, the first group in this case. Using this option results in 40 additional files that hold the calibrated and nodding-sutracted frames, the TWs, and the extracted spectra before demodulation.

5.3.6 Spectro-Astrometry

Spectro-astrometry uses the instrument derotator to take spectra with different position angles of the slit on the sky.

The recipe cr2res_obs_nodding can get raw input tagged like OBS_ASTROMETRY_OTHER or OBS_ASTROMETRY_JITTER. Given a sequence of these, it will separate them by position angle and reduce the set for each angle separately, just like plain nodding observations, with the same outputs. Any further steps are considered out of scope for the DRS and left to users.

[2]The rate and absolute distance have been measured manually and are hard-coded in the DRS.

5.4 Optional Steps

5.4.1 Throughput from standard stars

The pipeline comes with a fixed list of spectrophotometric standard stars. When `cr2res_obs_nodding` checks the on-sky coordinates of the currently reduced target, and if they match one of those stars, it compares the reduced spectrum to the reference. The resulting throughput gets saved into a separate data product (PRO.CATG=OBS_NODDING_THROUGHPUT).

Note however, that absolute flux calibration is generally not possible with CRIRES+ because of varying slit-losses.

5.4.2 Refining the BPM

Using only the BPM that comes from `cr2res_cal_dark` usually gives satisfactory results. However, pixels can be marked as bad also by being outliers in their non-linearity behavior, or in their overall sensitivity, i.e. in the normalized flat-field. The respective recipes each produce a BPM, and have parameters for the thresholds.

The recipe `cr2res_util_bpm_merge` allows you to merge these BPMs into a combined BPM; `cr2res_util_bpm_split` does the inverse, that is splitting a BPM into its original components. This is possible because these recipes keep track of why a pixel was marked as bad. When a BPM is then used in subsequent reduction steps, any kind of bad pixel gets masked and does not contribute to the result.

Note that BPMs are not checked to match in spectrograph setting, so that it is possible to use a BPM from a one setting with data from another. The same is true for DIT.

Experience has shown that the pair-wise subtraction via nodding makes many pixels usable that were flagged as bad pixels from the DARK frames. It can therefore make sense, especially for low S/N spectra, to use the BPM from from `cr2res_cal_flat` instead (`PRO.CATG=CAL_FLAT_BPM`). To do this, one needs to

- set the `--bpm_kappa` for `cr2res_cal_dark` to a large value (e.g. 1000) to get a master dark without pixels marked as bad. This is because otherwise the flat-field would inherit the BPM from the master dark.
- Adjust the parameters `--bpm_low` and `--bpm_high` for `cr2res_cal_flat` to appropriate relative sensitivity thresholds.
- The resulting `CAL_FLAT_BPM` can be used directly in subsequent reduction steps. However, it also marks the inter-order regions as bad which prevents them to be used for background subtraction via `--subtract_interorder_column` which defaults to `TRUE` for the obs-recipes.
- As a work-around, the BPM can be split into its components by feeding the `CAL_FLAT_BPM` to `cr2res_util_bp` which separates the BPM by the reason why a pixel was marked as bad. The number 2 signifies the outliers in the normalized flat-field, thus it is this BPM that can be passed to obs-recipes to allow inter-order background-subtraction.

5.4.3 Splicing and Continuum Normalization

Give extracted spectra, blaze functions and TWs from several settings to `cr2res_util_splice` and it will divide by the blaze, and resample the overlap regions between all spectra from individual detector-orders onto a common wavelength scale, in order to then combine them with a weighted average. The result is a single spectrum, continuous over the regions covered by the inputs.

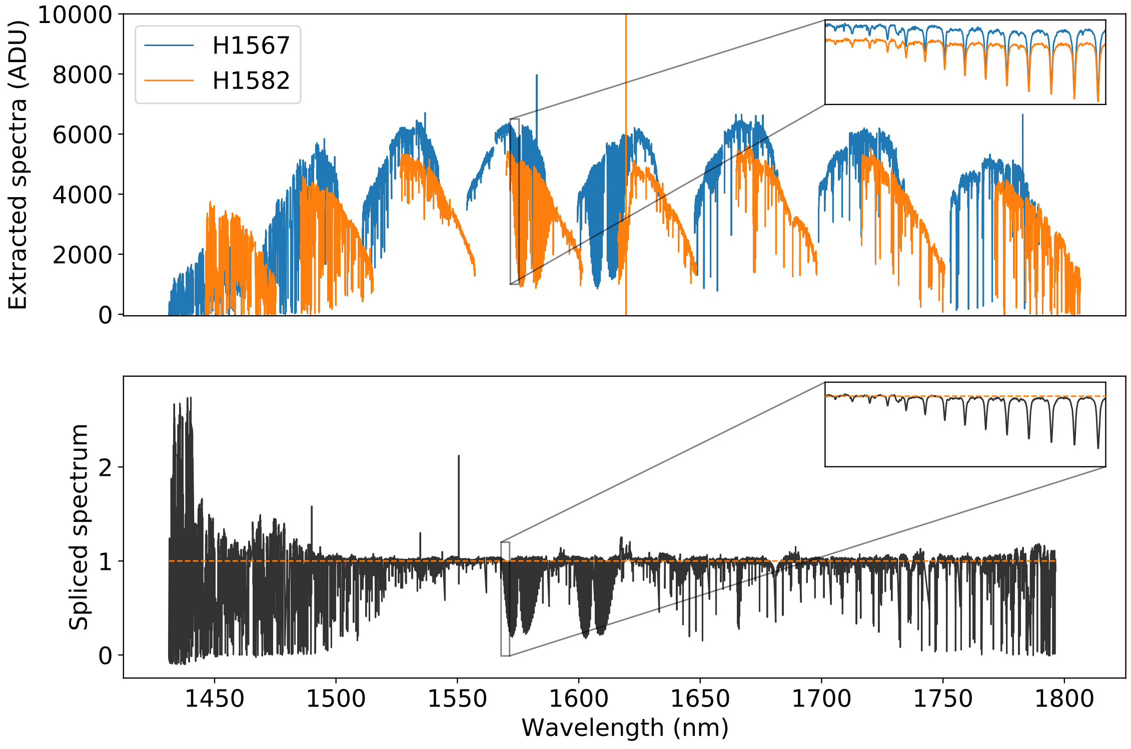


Figure 5.2: The top subplot shows extracted spectra in two overlapping wavelength settings: H1567 and H1582 with an inset zoom to a narrow wavelength region. The bottom subplot shows the output of the splicing recipe. Notice that the normalization is far from perfect.

Please note that this recipe has not been tested extensively to produce reliable results in all cases. For now its data products should therefore be considered experimental and quality not ensured to be "science-ready". Fig. 5.2 shows a typical example of what can be achieved with this recipe. An example SOF file is printed below.

```
1> cat splice.sof
H1567-cr2res_obs_nodding_extracted_combined.fits UTIL_EXTRACT_1D
H1582-cr2res_obs_nodding_extracted_combined.fits UTIL_EXTRACT_1D
$CALIB/H1567_tw.fits UTIL_WAVE_TW
$CALIB/H1582_tw.fits UTIL_WAVE_TW
$CALIB/H1567/05_cr2res_util_extract_out/
```

```
  cr2res_util_calib_calibrated_collapsed_extr1D.fits CAL_FLAT_EXTRACT_1D
$CALIB/H1582/05_cr2res_util_extract_out/
  cr2res_util_calib_calibrated_collapsed_extr1D.fits CAL_FLAT_EXTRACT_1D
```

It should also be noted that it is probably a good idea to manually mask some regions, like the ones affected by the ghosts, before splicing, since otherwise this wavelength region would be degraded in the output, even if it was unaffected in data from a different setting.

5.4.4 Blaze correction

Most of the recipes performing a 1D extraction (cr2res_obs_staring, cr2res_obs_nodding, cr2res_obs_pol) accept as optional input the blaze function computed by the cr2res_cal_flat recipe (PRO CATG = CAL_FLAT_EXTRACT_1D). If the file is provided, the extracted spectrum is divided by the normalised blaze function. The global normalisation factor is taken from a header keyword in the blaze file, if it was produced with version 1.6.0 or higher of the pipeline, otherwise the normalisation factor is the 95th percentile of the blaze for each spectral trace.

5.5 Calibration, from scratch

Following the steps outlined in Ch. 4.4.2 and accompanying text, we can reduce the calibrations without relying on pre-existing reduction products like the TW. This is also the way the static calibrations are assembled. The following gives example SOFs and recipe calls to reduce a single setting.

First the SOFs:

```
1> cat 01_cr2res_cal_dark.sof
CRIRE.2021-09-16T20:58:23.580.fits      DARK
CRIRE.2021-09-16T20:58:40.524.fits      DARK
CRIRE.2021-09-16T20:58:57.146.fits      DARK
CRIRE.2021-09-16T20:59:13.835.fits      DARK
CRIRE.2021-09-16T20:59:30.110.fits      DARK
CRIRE.2021-09-16T20:59:46.468.fits      DARK
CRIRE.2021-09-16T21:00:08.583.fits      DARK
CRIRE.2021-09-16T21:02:14.863.fits      DARK
CRIRE.2021-09-16T21:04:21.006.fits      DARK
CRIRE.2021-09-16T21:06:27.987.fits      DARK
CRIRE.2021-09-16T21:06:54.976.fits      DARK
CRIRE.2021-09-16T21:07:21.609.fits      DARK
CRIRE.2021-09-16T21:07:48.034.fits      DARK
CRIRE.2021-09-16T21:07:57.532.fits      DARK
CRIRE.2021-09-16T21:08:06.919.fits      DARK

2> cat 02_cr2res_util_calib.sof
../cr2res_cal_detlin_coeffs.fits CAL_DETLIN_COEFFS
CRIRE.2021-09-16T20:49:23.778.fits      FLAT
01_cr2res_cal_dark_out/cr2res_cal_dark_Y1028_3_bpm.fits CAL_DARK_BPM
01_cr2res_cal_dark_out/cr2res_cal_dark_Y1028_3_master.fits CAL_DARK_MASTER

3> cat 03_cr2res_util_trace.sof
02_cr2res_util_calib_out/cr2res_util_calib_calibrated_collapsed.fits    UTIL_CALIB
```

```
4> cat 04_cr2res_util_slit_curv.sof
CRIRE.2021-09-16T20:54:28.992.fits      WAVE_FPET
03_cr2res_util_trace_out/cr2res_util_calib_calibrated_collapsed_tw.fits  CAL_FLAT_TW

5> cat 05_cr2res_util_extract.sof
02_cr2res_util_calib_out/cr2res_util_calib_calibrated_collapsed.fits UTIL_CALIB
04_cr2res_util_slit_curv_out/cr2res_util_calib_calibrated_collapsed_tw_tw.fits UTIL_SLIT_CURV_TW

6> cat 06_cr2res_util_normflat.sof
02_cr2res_util_calib_out/cr2res_util_calib_calibrated_collapsed.fits UTIL_CALIB
05_cr2res_util_extract_out/cr2res_util_calib_calibrated_collapsed_extrModel.fits UTIL_SLIT_MODEL

7> cat 07_cr2res_util_calib.sof
../cr2res_cal_detlin_coeffs.fits CAL_DETLIN_COEFFS
CRIRE.2021-09-16T20:50:11.966.fits      WAVE_UNE
01_cr2res_cal_dark_out/cr2res_cal_dark_Y1028_20_bpm.fits CAL_DARK_BPM
01_cr2res_cal_dark_out/cr2res_cal_dark_Y1028_20_master.fits CAL_DARK_MASTER
06_cr2res_util_normflat_out/cr2res_util_normflat_Open_master_flat.fits CAL_FLAT_MASTER

8> cat 08_cr2res_util_extract.sof
07_cr2res_util_calib_out/cr2res_util_calib_calibrated_collapsed.fits UTIL_CALIB
04_cr2res_util_slit_curv_out/cr2res_util_calib_calibrated_collapsed_tw_tw.fits UTIL_SLIT_CURV_TW

9> cat 09_cr2res_util_genlines.sof
../lines_u_sarmiento.txt EMISSION_LINES_TXT
une_sel.dat LINES_SELECTION_TXT

10> # there is no step 10
11> cat 11_cr2res_util_wave.sof
08_cr2res_util_extract_out/cr2res_util_calib_calibrated_collapsed_extr1D.fits UTIL_EXTRACT_1D
04_cr2res_util_slit_curv_out/cr2res_util_calib_calibrated_collapsed_tw_tw.fits UTIL_SLIT_CURV_TW
09_cr2res_util_genlines_out/lines_u_sarmiento_une_sel.fits EMISSION_LINES

12> cat 12_cr2res_util_wave.sof
08_cr2res_util_extract_out/cr2res_util_calib_calibrated_collapsed_extr1D.fits UTIL_EXTRACT_1D
11_cr2res_util_wave_out/cr2res_util_calib_calibrated_collapsed_extr1D_tw.fits UTIL_SLIT_CURV_TW
09_cr2res_util_genlines_out/lines_u_sarmiento_une_sel.fits EMISSION_LINES

13> cat 13_cr2res_util_calib.sof
../cr2res_cal_detlin_coeffs.fits CAL_DETLIN_COEFFS
CRIRE.2021-09-16T20:54:28.992.fits      WAVE_FPET
01_cr2res_cal_dark_out/cr2res_cal_dark_Y1028_120_bpm.fits CAL_DARK_BPM
01_cr2res_cal_dark_out/cr2res_cal_dark_Y1028_120_master.fits CAL_DARK_MASTER
06_cr2res_util_normflat_out/cr2res_util_normflat_Open_master_flat.fits CAL_FLAT_MASTER

14> cat 14_cr2res_util_extract.sof
13_cr2res_util_calib_out/cr2res_util_calib_calibrated_collapsed.fits UTIL_CALIB
11_cr2res_util_wave_out/cr2res_util_calib_calibrated_collapsed_extr1D_tw.fits UTIL_SLIT_CURV_TW

15> cat 15_cr2res_util_wave.sof
14_cr2res_util_extract_out/cr2res_util_calib_calibrated_collapsed_extr1D.fits UTIL_EXTRACT_1D
11_cr2res_util_wave_out/cr2res_util_calib_calibrated_collapsed_extr1D_tw.fits UTIL_WAVE_TW
```

The commands to execute the steps are:

```
esorex --log-file=01_cr2res_cal_dark.log --output-dir=01_cr2res_cal_dark_out cr2res_cal_dark
  01_cr2res_cal_dark.sof
esorex --log-file=02_cr2res_util_calib.log --output-dir=02_cr2res_util_calib_out
  cr2res_util_calib --collapse="MEAN" 02_cr2res_util_calib.sof
esorex --log-file=03_cr2res_util_trace.log --output-dir=03_cr2res_util_trace_out
  cr2res_util_trace 03_cr2res_util_trace.sof
esorex --log-file=04_cr2res_util_slit_curv.log --output-dir=04_cr2res_util_slit_curv_out
  cr2res_util_slit_curv 04_cr2res_util_slit_curv.sof
esorex --log-file=05_cr2res_util_extract.log --output-dir=05_cr2res_util_extract_out
  cr2res_util_extract --smooth_slit=3 -smooth_spec=2
  05_cr2res_util_extract.sof
esorex --log-file=06_cr2res_util_normflat.log --output-dir=06_cr2res_util_normflat_out
  cr2res_util_normflat 06_cr2res_util_normflat.sof
esorex --log-file=07_cr2res_util_calib.log --output-dir=07_cr2res_util_calib_out
  cr2res_util_calib --collapse="MEAN" --subtract_nolight_rows=TRUE 07_cr2res_util_calib.sof
esorex --log-file=08_cr2res_util_extract.log --output-dir=08_cr2res_util_extract_out
  cr2res_util_extract --smooth_slit=30 --swath\_width=2048  08_cr2res_util_extract.sof
esorex --log-file=09_cr2res_util_genlines.log  --output-dir=09_cr2res_util_genlines_out
  cr2res_util_genlines 09_cr2res_util_genlines.sof
esorex --log-file=11_cr2res_util_wave.log --output-dir=11_cr2res_util_wave_out
  cr2res_util_wave --wl_method=XCORR --wl_degree=0 --keep --wl_err=0.1 --fallback
11_cr2res_util_wave.sof
esorex --log-file=12_cr2res_util_wave.log --output-dir=12_cr2res_util_wave_out
  cr2res_util_wave --wl_method=XCORR --wl_degree=2 --wl_err=0.03 --fallback
  12_cr2res_util_wave.sof
esorex --log-file=13_cr2res_util_calib.log --output-dir=13_cr2res_util_calib_out
  cr2res_util_calib --collapse="MEAN" 13_cr2res_util_calib.sof
esorex --log-file=14_cr2res_util_extract.log --output-dir=14_cr2res_util_extract_out
  cr2res_util_extract --smooth_slit=3 14_cr2res_util_extract.sof
esorex --log-file=15_cr2res_util_wave.log --output-dir=15_cr2res_util_wave_out
  cr2res_util_wave --wl_method=ETALON --wl_degree=4 --fallback 15_cr2res_util_wave.sof
```

A few notes on the recipe parameters used:

2. Remember, without `--collapse` the recipe would loop over the flat-field frames, dark-subtracting each, instead of doing so on the combined frame.

5. Extracting the flat-field frames, we know the slit is evenly illuminated, so we can apply some extra smoothing along the slit. The slight smoothing (regularization, actually) of the blaze spectrum is necessary because there are column-by-column variations in the detectors that would otherwise become part of the blaze, not part of the normalized flat-field, as intended.

7. For short exposures the detector background-level can vary significantly, so we use `--subtract_nolight_rows` to somewhat improve on suboptimal dark-subtraction.

9. The first pass of cross-correlation with a sufficiently large search-radius (`--wl_err`). `--wl_degree=0` means that only a shift in the zero-point is allowed, which means that we have to `--keep` the higher degrees of the input solution polynomial.

11. The UNE frames have very little signal in some detector orders, we therefore increase the smoothing along the slit and extract the full detector width in one swath, in order to increase the reliability.

12. It is possible to run second pass on the UNE wavecal with decreased search radius and increased polynomial degree. This is however *not* recommended any longer, since it is enough to get the zero-point from the UNE and leave the higher degrees to be determined from the FPET.

15. The numerous and regular lines of the FPET allow a polynomial with even higher degree.

9,11,15. Without `--fallback` the output would, in case of an error in a detector-order, not contain a solution for this order, thus precluding it from subsequent steps. With this option, the input solution gets propagated into the output, in case calibration fails.

6 Algorithms

6.1 General Algorithms

6.1.1 Optimal Extraction

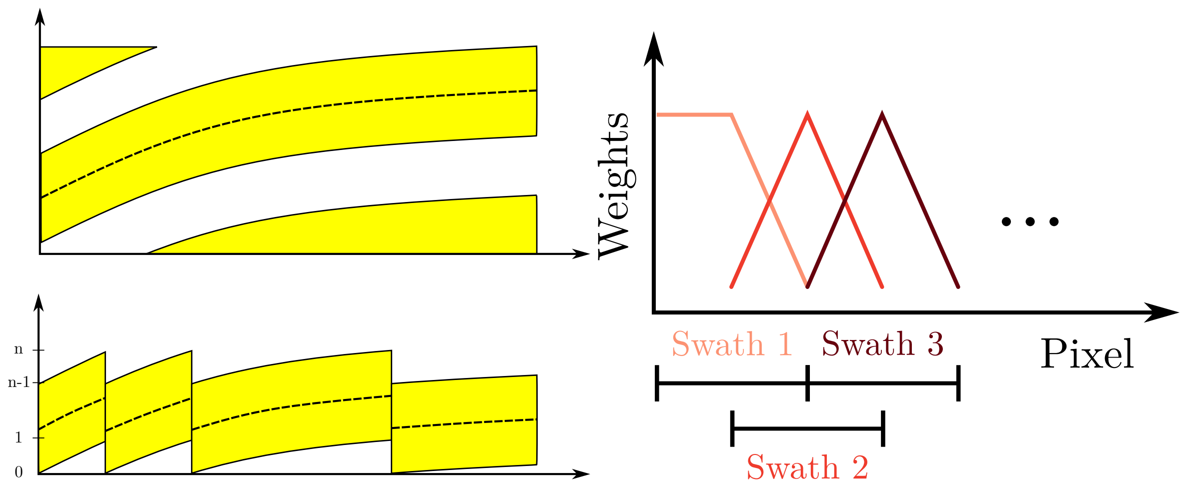


Figure 6.1: *Left: Illustration of swath rectification. Right: The weights for combining the spectra from overlapping swaths.*

The optimal extraction along the tilted and curved slit is a centerpiece of the CRIRES+ DRS and several recipes make use of it. Full details and mathematical description can be found in [6]. Here, we simply summarize the most important practical implications from a user perspective.

To extract a spectral order from a calibrated 2D frame into a 1D spectrum, using a *trace* from the TW table, the following steps are carried out:

- Calculate the extraction height from the edge-polynomials, if not explicitly given via `--extract_height`.
- Cut out a rectangular region around the mid-line, called a *swath* from the frame, using the height and the value of `--extract_swath_width`.
- Rectify the swath by shifting columns by integer values such that the mid-line only retains values between 0 ... 1, like illustrated in Figure 6.1 (left).
- Iteratively approximate the swath surface by two vectors: the spectrum along the mid-line and the slit-function along the slit projection, taking its changing tilt into account. The slit function is oversampled by a factor `--extract_oversample`.
- Step by a half swath-width to the next one, and repeat. This way swaths overlap fully and the order effectively gets extracted twice.
- Once all swaths are extracted, the spectra from each are merged into one, with linearly in-/decreasing weights, as in Figure 6.1 (right).

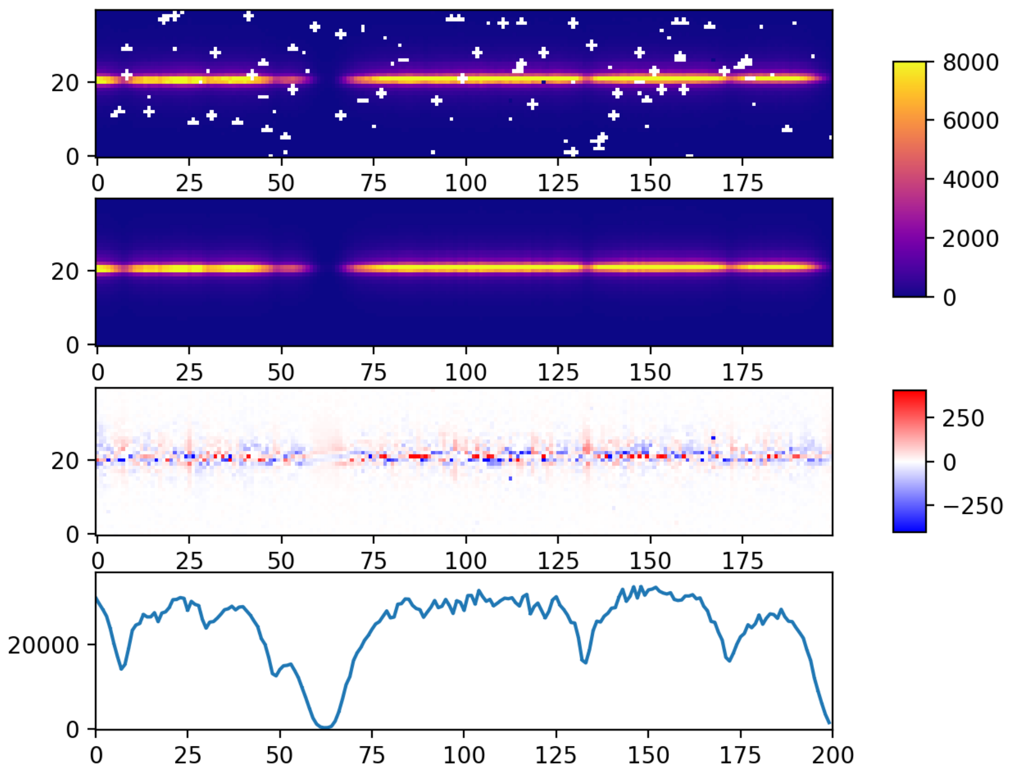


Figure 6.2: An example of the optimal extraction. *Top:* The flat-fielded, combined, and pair-wise subtracted frame, with the BPM applied and zoomed-in to part of a single spectral order. *Panel 2:* The model of the same region, reconstructed from the extracted spectrum and slit-function. *Panel 3:* The difference between the two above, showing the residuals. *Bottom:* The extracted spectrum itself, on the same x-axis as the panels above.

- The spectrum and slit-functions are saved, and used to reconstruct a 2D model of the frame, saved separately.
- The errors are estimated from the residual between data and model.

A major advantage of this extraction algorithm is that it makes no assumption about the slit-function, except that it does not change within a swath. This allows, for example, to combine jittered nodding frames into one, and extract it together.

In addition, the model by design cannot approximate features that are not present in all pixels that contribute to a certain bin in the spectrum or the slit-function. This makes spectra robust towards e.g. cosmic rays.

Fig. 6.3 shows the effect of the *slit-function oversampling*, that can be changed via the parameter `--extract_oversample`. As mentioned before, this factor linearly affects the computation time. The amount of oversampling necessary depends on how quickly the mid-line of a spectral order changes with respect to detector rows, and how sharply the slit-function changes. The default value of 5 is enough in many situations, but clearly leaves artifacts in the example shown, which was chosen to be close to worst-case, and requires an oversampling > 12 for optimal results.

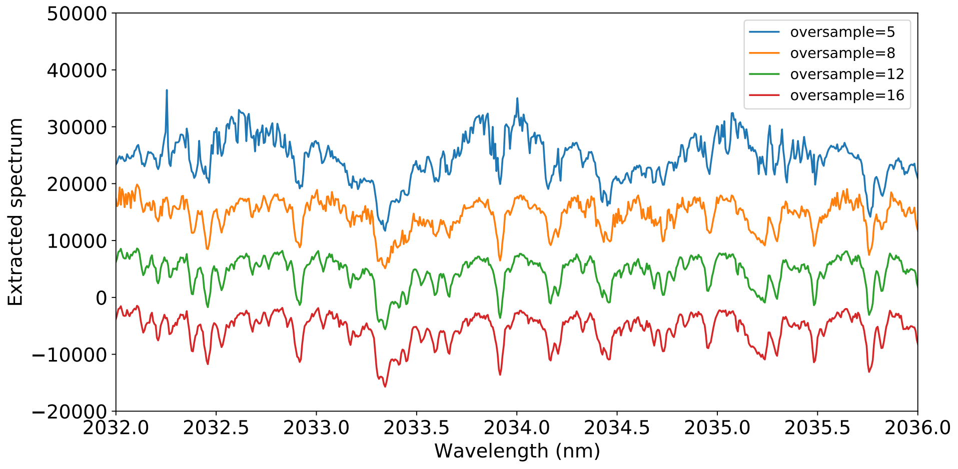


Figure 6.3: Example spectrum, extracted with varying oversampling, and vertically offset for plotting. In this case, oversampling of 12 is needed to no longer see "spiky" artifacts.

There are a few more parameters that influence the extraction:

- `--extract_swath_width` sets the width of the swath, in pixels, i.e. how large a piece of a spectral order should be processed at a time. Meaningful values range from 50 (high S/N necessary) to the full detector width. The default is 800 and works in most cases, because the slit-illumination rarely varies on that scale.

- `--extract_height` sets the height in pixels of the extracted region, symmetrical around the mid-line of the trace. By default, this is not set and the upper and lower edge polynomials of the trace are used to set the height. Setting this parameter therefore overrides the role of the edge polynomials in the extraction. In other words, this parameter takes precedence over the slit-fraction when it comes to specifying the extraction height.

- `--extract_smooth_slit` is a regularization parameter for the slit-illumination vector, as it is approximated during the extraction. The value is roughly the size of the smoothing kernel in pixels, and should not be set below 1.0 to avoid ringing. Default is 2.0.

- `--extract_smooth_spec` is analogous to the previous, but along the spectrum instead. It defaults to 0.0 to not degrade resolution, but can be increased as needed.

For the utility recipe `cr2res_util_extract` these parameters are named without the prefix `extract_`.

6.1.2 Other extraction methods

`cr2res_util_extract` has another feature up its sleeve: the parameter `--method` which allows using different algorithms to extract spectra from the 2D frame. It can take the following values.

- `SUM`, a simple vertical sum of the spectral order. Because of the non-vertical slit, spectral features will be broadened.
- `MEDIAN`, same as previous, but calculating the median instead of the sum. The value is multiplied by the extraction height, so that the normalization is the same as the other methods.
- `TILTSUM` is like `SUM`, but first shifts the detector rows according to the slit tilt. Only integer shifts are applied, so there is still smoothing from sub-pixel tilt.
- `OPT_CURV`, the default optimal extraction, as described above.

For all other recipes, `OPT_CURV` is the only choice.

6.1.3 Errors and Signal-to-Noise

Raw frames contain pixel values in ADU. All data products are in ADU also, i.e. no conversion to electrons is made, and no normalization to $1s$-exposure by dividing by `DIT`. There are however several steps along the way to the final spectra that affect the absolute values of pixels and spectral bins, which is why it makes sense to summarize them here.

Some things to keep in mind about ADU values

- For data with `NDIT >1`, each sub-exposure gets averaged in the detector software.
- In a nodding sequence the pair-wise subtracted frames are also averaged.
- The normalization of the output-spectra is like that of a sum of pixel values, collapsed along the slit, i.e. summed-up ADU, also for the extraction methods `OPT_CURV` and `MEDIAN`.

Errors, both in maps and spectra are absolute. Calibrations are applied to raw frames in the following order, at each step propagating the error into the output

- Bad pixel map[3]
- Detector non-linearity.
- Shot-noise gets added to each pixel's error, based on $\sqrt{\texttt{ADU}}$, divided by $\sqrt{\texttt{NDIT}}$ and $\sqrt{\texttt{GAIN}}$.
- Read noise gets added to each pixel's error, following the description given in the CRIRES+ manual.
- Dark-subtraction. The master darks are DETLIN-corrected, because they are not "dark" but contain thermal emission.

[3]BPMs have no errors, therefore no error-propagation at this step.

- Optional subtraction of all detector rows with a row, that is derived by taking the vertical median over the 40 bottom rows which receive no light.
- Division by the normalized flat-field.
- Optional correction for cosmic rays.

N.B.: ESO is aware that the method described below often underestimates the errors. This issue will be analysed and addressed in a future version of the DRS. For now three approaches have been implemented in `cr2res_obs_nodding` to aid the user in estimating the errors.

- A DER SNR keyword is added to the FITS extensions containing the extracted spectra of the form `ESO QC DER SNRn`, where n is the order number. This is an estimate of the average S/N in the whole trace following the method described in [11]. Due to the complex nature of many CRIRES+ spectra, this begins to lose accuracy at high S/N, but values below 70 should be accurate to at least 25%.
- For IDP products, produced using the `--idp` flag, a correction term is added to the errors to bring the result closer to the S/N derived by the DER algorithm.
- Using the parameter `--error_method=Horne`, the variance is calculated using the $\mathrm{var}\,|f| =$... formula from equation 8 in Horne [5], all input parameters are taken from the standard DRS extraction method. This method should provide relatively robust results, and will include contributions from thermal background and telluric emission lines, at very high S/N it may underestimate the errors, as it does not include systematics or covariance.

Note that none of the above methods have been fully verified as yet.

For the optimal extraction the accumulated signal strength in each spectral bin is assumed to be dominated by electron shot noise (Poissonian), taking the various multipliers (NDIT, number of averaged frames) and the detector gain into account. [4]

Note that to and including version 1.3 of the DRS, the errors were instead based on the *residual* of the extraction model. This frequently led to an underestimation of the signal-to-noise ratio (S/N), therefore the errors and S/N will *change* when re-running previous reductions with the latest version of the DRS.

Outliers (e.g. cosmics) that do not affect the extracted spectrum much, previously increased the corresponding error unnecessarily; this is no longer the case. On the other hand, the model residual is very large when the extraction model cannot accommodate the data, like in the regions affected by ghosts, which is why the ghost-regions had large errors before, but no longer do to the same extent.

With the new method based on accumulated signal, the error can become undefined (NaN) when a spectral bin has close to zero signal, e.g. in saturated absorption lines.

The spectrum can be directly divided by the error-spectrum to obtain the S/N; no further conversions are necessary. In order to manually compare this S/N with the theoretical expectation from Poissonian statistics, users will have to scale the ADU-values of the spectrum by the gain and the number of averaged frames (NDIT or combined frames), before taking the square-root (Eq. 1).

[4]Up to and including version 1.3 of the DRS, the errors were instead based on the *residual* of the extraction model. This frequently led to an underestimation of the signal-to-noise ratio.

$$S/N = \sqrt{(ADU_A + ADU_B) * NDIT * NEXP * NABCYCLES * GAIN} \quad , \tag{1}$$

where $ADU_{A,B}$ are the spectra from nodding positions A and B, respectively. For other types of observations replace accordingly. Polarimetric observations always take four exposures at different SPU-angles; the demodulated intensity spectrum (Stokes I) therefore needs an additional factor 4 in the equation.

Note that an important factor in getting high S/N in spectra is to apply deep enough flat-fields, that ideally exceed the S/N in the observations. And that the detector linearity correction can be a limiting factor, see below.

6.2 Recipes Algorithms

6.2.1 Order Tracing

As written above, by a *trace*, we mean a polynomial, in pixel-coordinates of a single detector, that follows the mid-line of a spectral order. And two more polynomials for the upper and lower edges of the slit projection, as shown in Fig. 4.2.

The way this is derived is by first determining which pixels in a FLAT frame contain signal, and which don't. To that end the image is smoothed first horizontally, then vertically, and then compared to the un-smoothed frame via thresholding. The recipe option `trace_opening`, true by default, joins together closely adjacent clusters of pixels.

After that it is determined which spectral order the pixels belong to. For this purpose, the mean Y-coordinate for each order is present in the raw file headers.

Lastly, the X and Y-coordinates for pixels within the same order are fitted with a polynomial, which gives the mid-line of the trace. In addition, edge-detection gives the upper and lower edges of the spectral order.

6.2.2 Measuring the slit-shape

The recipe `cr2res_util_slit_curv` takes as input a TW, for the order traces, and an FPET frame. The algorithm finds the etalon peaks, which give good coverage in each spectral order. It then fits the shape of each peak with a polynomial P(y), because in principle a curved slit could cross the same x-coordinate more than once.

The coefficients of these polynomials are then fitted with a P(x) for each degree. This ensures a smooth change of the slit tilt with wavelength, removes outliers, and allows to store the information in the TW in an analogous way to the polynomials for trace and wavelength solution. The TW-columns are named `SlitPolyA`, `SlitPolyB` and `SlitPolyC`.

As of DRS version 1.2.4 (Feb 2023), two changes are introduced to the way the slit tilt and curvature are handled:

1. The slit shape is no longer *static*, in the sense that the reductions using the `cr2res_cal_*` recipes or the Reflex workflow not longer rely on the values in the static TW-files. Instead the recipe `cr2res_cal_wave` now starts by measuring the slit shape and saves the result together with the wavelength calibration in the

output TW. This change is because the slit tilt has unexpectedly been found to change after a warm-up cycle.

2. The choice between a straight slit or a parabolic curve is no longer hardcoded but offered as the parameter `--degree` or `--slit_degree` on recipes `cr2res_util_slit_curv` and `cr2res_cal_wave`, respectively. The default has been changed to second degree, i.e. a parabolic slit, which has been found to give slightly better residuals after extraction, even though the slit is much more tilted than curved.

6.2.3 Wavelength calibration

There are several ways to calibrate the wavelength scale, and each corresponds to one of the values that the parameter `--wl_method` of `cr2res_util_wave` can take.

With `--wl_method=XCORR` the input catalog of lines is converted into a synthetic spectrum that is cross-correlated with the extracted lamp spectrum. The first-guess solution is varied within the rage allowed by the parameters `--wl_degree` and `--wl_err`, also taking a `--wl_shift` into account, if given. The result with the best cross-correlation outcome is kept as result. Each detector-order is calibrated separately.

The special case of `--degree=0` means a shift of the zero-point only, without changing the dispersion or the higher orders of the input solution. Therefore `--keep=TRUE` is needed, which does exactly that. Without it, the output solution always has the degree that is asked for via `--degree`, and there cannot be a 0th-degree solution.

By the way, the option `--fallback`, when TRUE, also allows you to propagate an input solution into the output, but under a different condition. Namely when the calibration of a detector-order fails.

The calibration spectra that are used for the cross-correlation are from an Uranium-Neon lamp (UNE). The catalogs for U-lines come from [9] and [7]. They are used for YJ and HK-bands, respectively. Note that all wavelengths are vacuum, not air.

There are two other methods, not used by default, that can be applied to the UNE lamp data, `LINE1D` and `LINE2D`. Instead of cross-correlating a synthetic spectrum, these go through the list of catalog lines and try to fit each with a Gaussian. The line-centers found this way are then fit against the catalog wavelength to arrive at the solution. This method is naturally more sensitive to blended lines and depends on the catalog line being the strongest peak in the spectrum cut-out that is fitted, which in turn is determined by the input solution and `--wl_err`. The catalogs with lines for each setting that are distributed with the DRS are selected with the XCORR in mind, because in some spectral regions lines are sparse enough that weak lines have to be used. Therefore it is likely necessary to make a further, or altogether different selection of lines to get reliable results with line-fitting.

The difference between `LINE1D` and `LINE2D` is that the former treats each detector-order independently, like `XCORR`, while the latter makes use of the grating equation that connects the solution between orders. The absolute order number becomes the Y-coordinate of the 2D-polynomial and the parameter `--xdegree`[5] sets the degree in this direction.

The method used for calibrating with the spectra from the FPET is `--wl_method=ETALON` and it is heavily inspired by [RD05][1]. It makes use of the intrinsic regularity of the FPET peaks, and applies a 2D-fit to the

[5]The `x` in the name stands for "cross-dispersion", not the X-coordinate.

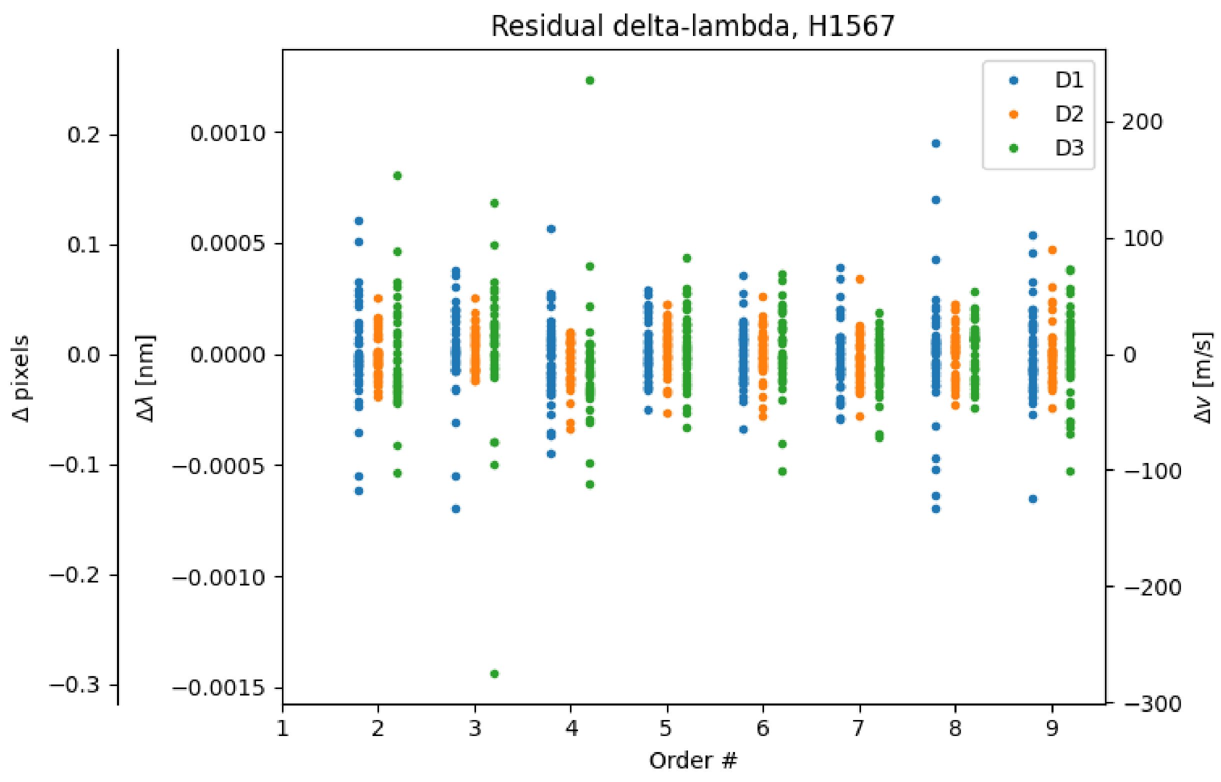


Figure 6.4: A representative example of residual values after wavelength calibration with the FPET. Each dot indicates the difference between expectation and reality, for the final wavelength solution that was found. Axes in several units are included.

spectra from each detector. More details in a future iteration of this document.

In contrast to `cr2res_util_wave` which executes a single wavecal-method on pre-processed data, the recipe `cr2res_cal_wave` works on raw wavecal inputs and executes several calibration steps after one another, using both UNE and FPET as taken together by the wavecal template. (If only one or the other is supplied, then naturally only the appropriate method is executed.)

The calibration strategy could, if one were inclined to do so, be replicated with individual calls of util-recipes, like this:

- apply master calibrations to raw (UNE and FPET) frames, with `cr2res_util_calib`
- extract spectra from the calibrated frames, `cr2res_util_extract`
- calibrate the UNE spectra with `--wl_method=XCORR`, `degree=0` and `--keep`, which means a shift of the zero-point. `cr2res_util_wave`
- calibrate the extracted FPET spectra, using the solution from the previous step as starting point. `cr2res_util_wave --wl_method=ETALON --degree=4 --xdegree=6`

This two-step strategy has been found to give reliable results in all instrument settings in Y-K bands. An example of final residuals is shown in Fig. 6.4.

As noted in 7, there is no active wavelength-calibration for the L and M-bands, since neither the UNE or the FPET are usable there. The use of gas-cells is being investigated, but in the meantime, using `molecfit` has been found to give good results, see App. C.

Also note that the TW tables that are delivered with the DRS static calibrations already have wavelength solutions derived this way, and should therefore be readily usable for observations taken with metrology.

6.2.4 Background Subtraction

In general, the best way to subtract background – be it originating from the sky, telescope, instrument or detector – is to use nodding in observations. As explained above, the recipes take care of the frame subtraction and combination.

There are two more ways to subtract background which can be activated with the following parameters of the appropriate recipes.

The bottom 40*pix* rows of the detectors are physically blocked from receiving light. When `--subtract_nolight_rows=TRUE` then the DRS will take a vertical median over these rows, and then subtract the result from all detector rows.

With `--subtract_interorder_column=TRUE` given, recipes will fit a low-order polynomial to the pixel values between the spectral orders[6], and then subtract the result. This is done for each column independently which allows it to address both scattered light and mitigate the occasional detector artifacts that randomly affect a certain vertical read-out channel.

The latter method has given good results in tests and is therefore turned on by default. We cannot however completely exclude the possibility of negative side-effects in certain situations.

6.2.5 Detector non-linearity

Users are provided (via the ESO Archive[7]) with the calibration data-product (`PRO.CATG=CAL_DETLIN_COEFFS`) that is needed to apply the correction for the non-linear response of detector pixels. This file contains for each pixel the coefficients of a polynomial which, when evaluated at a certain ADU-level, yields the correction factor by which to multiply the raw ADU, so that the value becomes that which a linear response would have given.

The `cr2res_obs_*` recipes take this file as optional input and apply the correction, as do some calibration recipes. Keep in mind however that the 0.5% error of each pixel's correction effectively **limits the S/N** of the resulting spectra. The detlin should *not* be applied to the flat-field because doing so introduces the noise a second time. Methods to reduce the detlin noise are being investigated.

[6]The master-flat is used to determine these pixels, because at this stage of the frame calibration the DRS does not have access to a TW table. Therefore, this method requires a flat-field to be present in calibration inputs.

[7]As of writing (2022-02-22), this reduced calibration is not yet in the Archive, but should become available in the coming weeks. If 20GB of download and disk space are not an issue, then feel free to search for the raw data via `TPL.ID=CRIRES_spec_tec_DetLinMon` and run the recipe yourself.

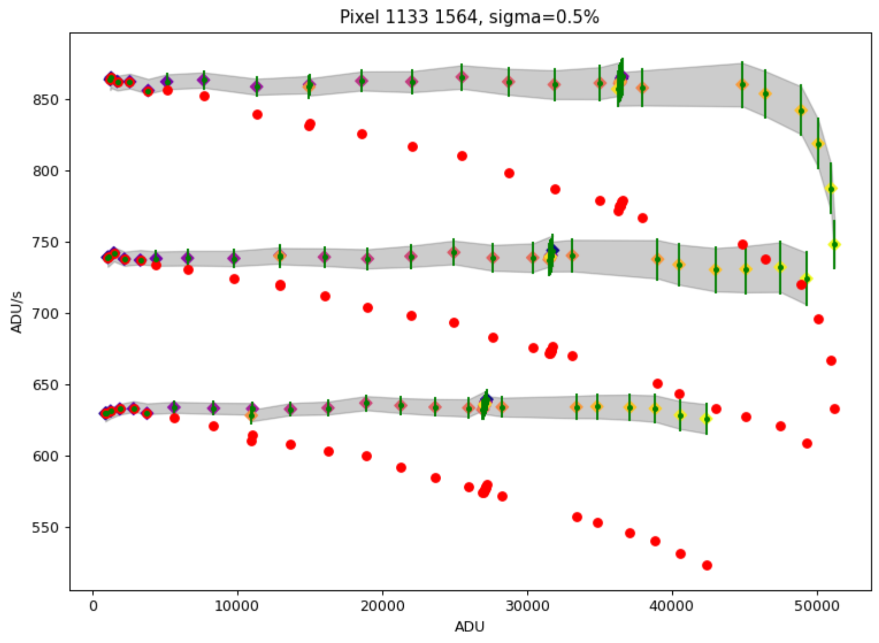


Figure 6.5: Before and after non-linearity correction for a randomly picked pixel in each detector. Red circles: Raw ADU/s vs. ADU. Green dots with errorbars: ADU/s as read from FLATs to which the previously derived calibration has been applied, using `cr2res_util_calib`. The closer to horizontal the points fall, the better the correction, because the actual illumination is kept constant. The standard-deviation from 1 is given in the title. Diamonds and shaded region: As a cross-check, the plotting-script separately reads and applies the correction to the raw values; should align with green dots and errors.

In order to derive this correction, a long series of exposures with the flat-field lamp are taken by the aforementioned template, with increasing DITs, starting from very short until well into saturation. This is repeated for several spectrograph settings in order to cover the gaps between the spectral orders. A series of DARKs with matching DIT is also obtained and subtracted from the FLATS, because the non-zero detector background pattern at short DITs would otherwise influence the determination of the linear response.

The reduction of the data set is done by `cr2res_cal_detlin`

```
1> cat detlin.sof
CRIRE.2021-10-18T09:50:30.827.fits DETLIN_LAMP
CRIRE.2021-10-18T09:50:32.845.fits DETLIN_LAMP
...
CRIRE.2021-10-18T09:59:12:345.fits DETLIN_DARK
...

2> esorex cr2res_cal_detlin detlin.sof
```

For each pixel, the recipe carries out the following steps on the sequence of images:

- Convert ADU to ADU/s by dividing by DIT. Plotting ADU/s versus ADU gives a horizontal line in the linear regime (illuminating flux is kept constant), and deviates downward for larger ADU (Fig. 6.5).

- Assume linear response up to 4000 ADU and take median of all measurements below this threshold to find the *true ADU/s*.
- Divide the non-linear ADU/s by the true value to get the correction factors, depending on ADU-level.
- Fit a parabola to the correction factors' dependence on ADU.
- Save the coefficients into data product.

This procedure is repeated for the other spectrograph settings and in the end the coefficients are merged, with weights according to their errors.

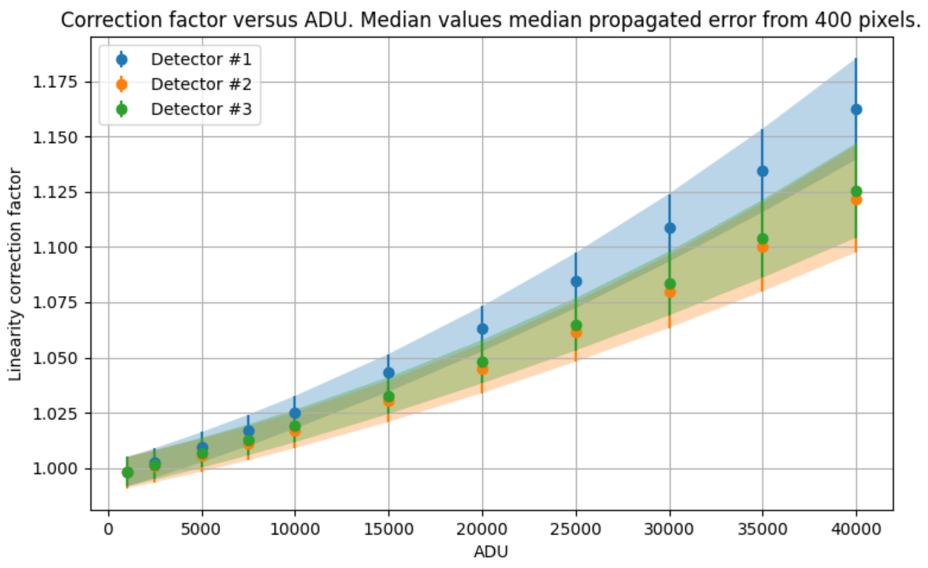


Figure 6.6: The size of the non-linearity correction, depending on raw ADU-level.

The additional figures 6.7, 6.6 and 6.8 show some statistics of the outcome. For example, one way to read these figures is that at 30,000 ADU, a correction of around 10% needs to be applied, and this correction is accurate to around 0.5%.

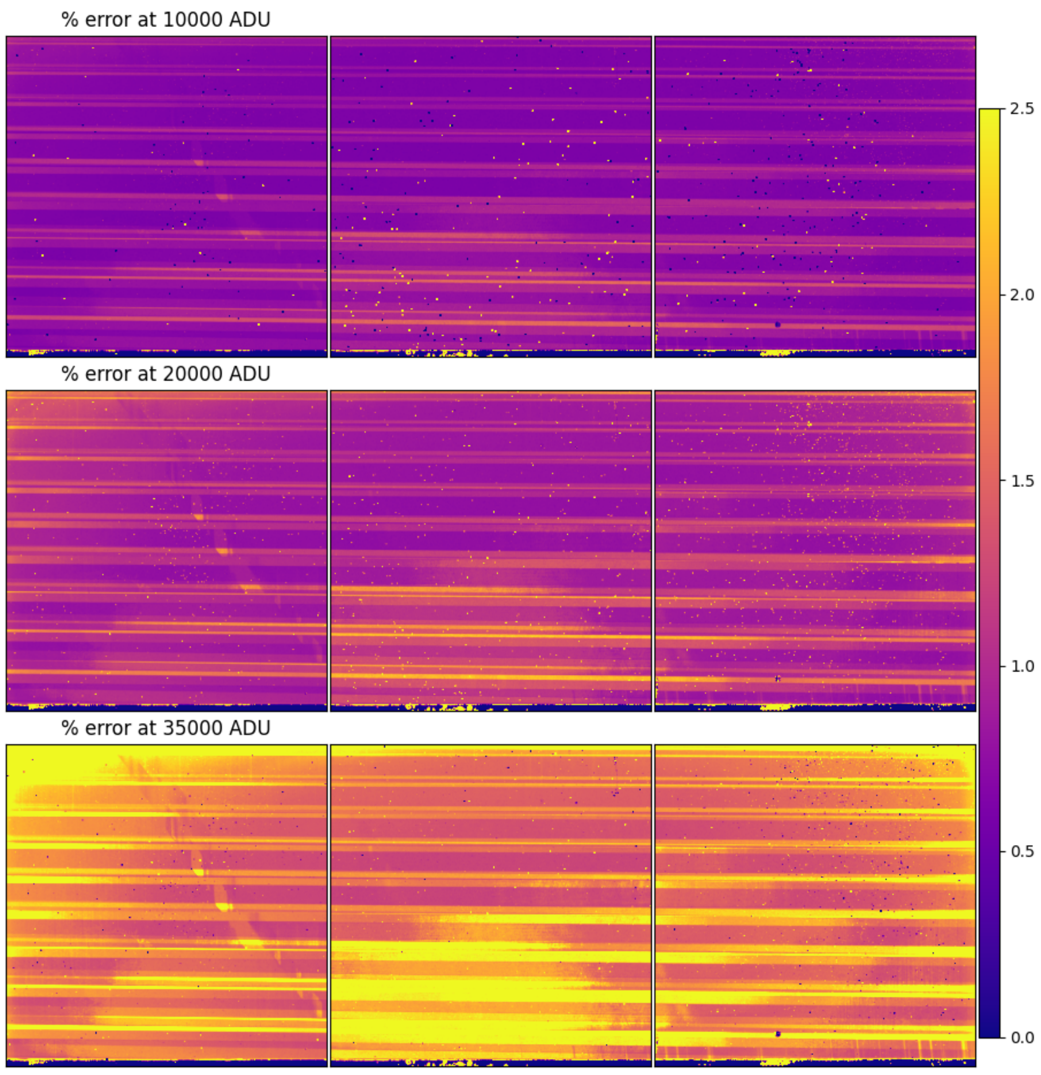


Figure 6.7: Error maps for three ADU levels. These are the propagated errors of the correction factor for each pixel, at that ADU. Clearly visible are the spectral orders and gaps, with the largest errors for the pixels that are only illuminated by one of the three spectrograph settings.

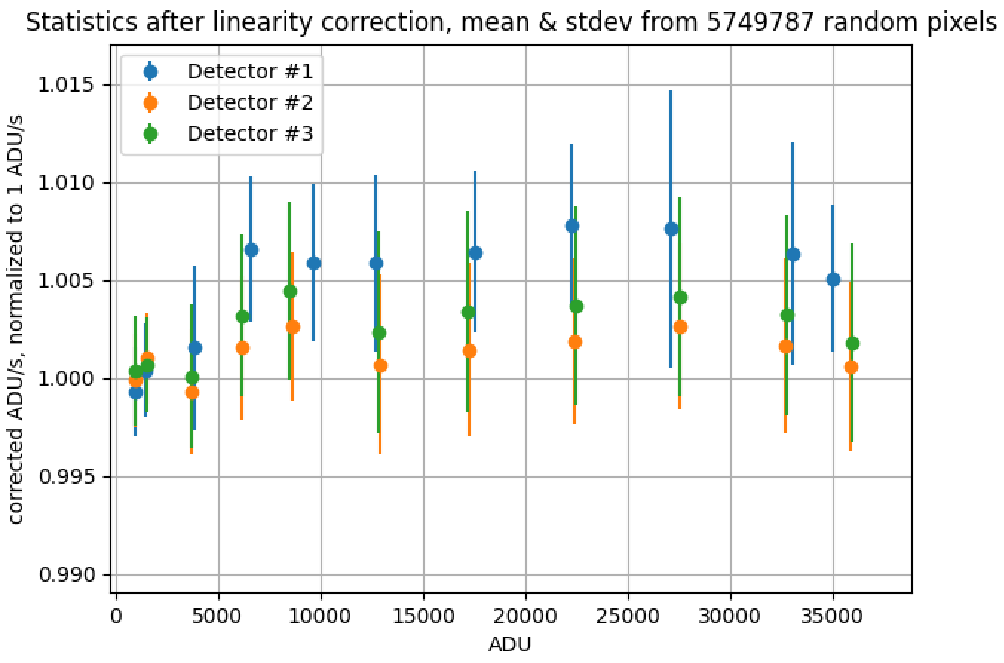


Figure 6.8: How well does a known constant illumination get recovered by applying the correction? This is what the deviation from the horizontal line at 1.0 in this figure shows. A value of 1.005 on the y-axis corresponds to 0.5% inaccuracy. Points are average deviations of random pixels for a certain ADU-bin; errorbars here are the standard-deviation of values in the same bin, not the propagated errors.

7 Known Issues

7.1 Superresolution

In good seeing conditions the AO system can produce a PSF that is sharp enough to not evenly fill the slit width of the spectrograph. While this produces spectra with a resolution higher than specified, it can also cause an offset in the wavelength scale, because how the AO places the PSF center is not reproducible enough. Wavelength shifts between reduced spectra from the nodding positions A and B of around 1*pix* have been encountered. [8]

This is not something that the DRS can correct for and will need to be mitigated by allowing users to choose a "slit scanning" AO mode, in which the PSF center is moved by small amounts between the slit edges during exposures.

Users can correct the wavelength offset between A and B spectra relatively easily, by cross-correlating one to the other before combining them.

The following warning will be issued by the recipes `cr2res_obs_nodding` and `cr2res_obs_staring` when it looks like this is the case, judging from the illumination profile *along* the slit.

```
Median FWMH of the PSF along the slit
is NN pix, i.e. below the slit width. This means the slit
is likely not evenly filled with light
in the spectral direction. This can result in a
wavelength offset between different postitions along the slit,
 and with respect to calibrations.
```

7.1.1 . . . in polarimetry

Although we are not aware of a data set that exhibits this, the wavelength-shift due to superresolution can be expected to affect the two beams of the polarimeter. The result after demodulation would be a non-zero polarization signal in all spectral lines.

Such data would need more manual effort in data reduction, namely extracting the beams one at a time and cross-correlating the spectra before demodulating.

7.2 Calibration caveats in L and M band

This is not a DRS-issue per se, but a current limitation of the instrument calibration in general. The UNE and FPET cover only the bands YJHK, which means that in L and M

- the slit-tilt is not characterized, and spectrum extraction happens along a vertical slit.
- there is no wavelength calibration within the DRS. Instead, the wavelength scale from static calibrations (derived manually from telluric lines), is propagated to the data products.

[8] This is not to be confused with the totally normal spectrum offset due to slit tilt, see 8.1

7.3 BPM in L and M

The strong thermal emission present in the DARKs in L and M band makes the detection of bad pixels more difficult than in the shorter bands. It makes sense therefore to compare the derived BPM with one from e.g. Y-band and vary the parameter `--bpm_kappa`.

Simply using the BPM from another band has also been found to work well, cf. 5.4.2.

8 Frequently Asked Questions

8.1 Spectra are shifted between nodding A and B

As mentioned throughout, the CRIRES+ slit is not aligned with detector columns, which means that spectra from different positions along the slit will have different wavelength scales. The DRS takes care of this when assigning wavelengths to spectral bins, for example by calculating new wavelength solutions for the nodding positions A and B, using the slit tilt calibration. Therefore, features in spectra from A and B should align with each other when plotting against wavelength (unless 7.1), but *not* when straight-up plotting the spectral bins against their index (i.e. pixels).

8.2 Overlapping spectra in IDP

For the IDP data format (cf. 4.1.5), the spectra from all detector-orders in a setting are stored in single long arrays for wavelength, ADU and errors, sorted by wavelength.

This naturally leads to large gaps in the spectrum, between detectors and orders. In the Y-band, there is also some wavelength-overlap between orders and this, in combination with the fact that spectra are not normalized, results in large “jumps” in the spectrum in the overlap region, when plotting outright.

The solution is to use the additional data column, which gives the order number for each spectral bin, to separate the overlap regions. Only after normalization can the spectra be merged or spliced together, e.g. using `cr2res_utiul_splice`.

8.3 Barycentric correction

As of late 2023, the DRS provides a value for the barycentric velocity correction. This is not specific to CRIRES+ but uses the appropriate functionality from the HDRL. For details on the algorithm and its inputs, see the documentation of the ESO Toolkit Library (ESOTK)[9].

The velocity offset is saved to the data products’ header key `DRS BARCORR`, and also printed in the logs.

The correction is not applied to the wavelength scale of the spectra.

8.4 Timestamps

For observations that comprise a time-sequence, e.g. an exoplanet transit, the timestamp of reduced data is relevant. The DRS does currently not calculate the (weighted) average timestamp of exposures that are combined. This “midpoint” in time will get added as a separate header key in the future.

In general, data products inherit their FITS headers from the *first raw frame* in the input SOF, this includes `OBS.MJD`. Of course, the timestamps of the raw frames allow users to carry out the calculation manually.

[9] https://eso.org/sci/software/pipelines/esotk/esotk-pipe-recipes.html

9 Troubleshooting

9.1 Checking the TW

In case the extracted spectra do not match the expectation from the raw frames, the first thing is to check the traces in TW, i.e. if the correct region gets extracted.

This can be most easily done by plotting the traces on top of a frame, either a raw frame, or an intermediate product like the nodding-combined image with positive and negative target spectra from positions A and B, respectively. Note that `cr2res_obs_nodding` also saves the traces that it derived for positions A and B.

Such a plot can be made with `cr2res_util_trace_map` or one of the scripts from App. **??**.

Similarly, the slit-tilt in the TW should be checked if the resolution of spectra is lower than it should be.

9.2 Imperfect Order Traces

The tracing of spectral orders can be less reliable when telluric absorption is very strong, like in some orders in L and M-bands. The parameters of `cr2res_util_trace` allow to change the amount of smoothing in x and y-direction, and the threshold for signal-detection. Varying these is the first step to improve the traces.

Recipes that extract spectra into 1D have the parameter `--extract_height` which allows you to ignore the edge-detected upper and lower polynomials and instead use a fixed extraction height, in pixels, centered on the mid-line. This can be used as a work-around when the mid-line of a trace is fine, but the edges are not.

If for some reason the traces cannot be derived from FLATs at all, `cr2res_util_trace` can be used on the science frame itself. This works well if the target has a reasonably strong continuum.

9.3 Data with low S/N

For data with low signal-to-noise ratio, and sources with low continuum emission, the *swath width* can be increased, in order to allow the algorithm to get a good estimate of the slit-illumination function. It is even possible to extract each detector-order as a single swath(`--swath-width=2048`).

Furthermore, the parameter for smoothing (regularization) along the slit can be increased to match the data, e.g. atmospheric conditions and (non-)AO. The corresponding parameter to regularize in the spectral direction can be used to smooth to the slit-width (2.5 or 5 pixels), or beyond, if the full spectral resolution is not needed for the science.

9.4 Artifacts in 1D spectra

In general, the optimal extraction is robust towards local outliers, as long as they do not affect all pixels that contribute to a spectral bin. When this is the case though, like a cosmic ray that hit a large fraction of the extraction height, it can be beneficial to mask out the cosmic first. This can be done by setting `--cosmics=TRUE` when running the `cr2res_obs_*` recipe.

In DRL versions prior to 1.3, cosmics could in rare cases cause an extraction swath to converge wrongly, resulting in swath-width-sized, triangularly shaped depressions in the spectrum. If you encounter this in your spectra, we advise to re-reduce with the latest version of the DRS.

9.5 Speeding up

The optimal extraction that all `cr2res_obs*` recipes (except `cr2res_obs_2d`) use is computationally expensive. The easiest way to save time is to shorten the *extraction height* to not include unnecessary parts of the slit.

For example, if the order trace matches your target (see above), no jitter was used and the seeing/AO conditions were good, the height can be set as low as 20 pixels. We recommend however to check the slit-function and/or the extraction model to make sure, no part of the target PSF is being cut off.

Cf. 5.3.2 for how to extract a specific part of the slit.

9.6 Debug Settings

The environment variable `ESOREX_MSG_LEVEL` controls how much logging output is produced, and can take the values `off`, `error`, `warning`, `info` and `debug`. If it is set to `debug`, the DRS not only produces extra logging output, but also dumps some intermediate output into FITS files named as `debug_*.fits`. These are not documented, but can be self-explanatory.

Instead of the environment variable, the message level can also be set as command option:

```
1> esorex --msg-level=debug ...
```

10 Recipe man-pages

The following are auto-generated chapters that contain the manual-pages of each recipe. These can be viewed at any time on the command-line with

```
esorex --man-page <recipe>
```

where `<recipe>` stands for the recipe name.

In order to avoid duplication of information, this serves as reference for for inputs, outputs and parameters of each recipe. Explanations and context for algorithms and non-obvious parameters are provided in the appropriate chapters.

10.1 cr2res_cal_dark

```
NAME
  cr2res_cal_dark -- Dark recipe

SYNOPSIS
  esorex [esorex-options] cr2res_cal_dark [cr2res_cal_dark-options] sof

DESCRIPTION
  Dark
    Compute the master dark

    Inputs
      raw.fits DARK [3 to n]

    Outputs
      cr2res_cal_dark_[setting]_DITxNDIT_master.fits CAL_DARK_MASTER
      cr2res_cal_dark_[setting]_DITxNDIT_bpm.fits CAL_DARK_BPM

    Algorithm
      group the input frames by different values of DET SEQ1 DIT
                  or/and DET NDIT or/and WLEN ID
      loop on groups g:
        loop on detectors d:
          Load the images and create the associate error for each of
                  them using cr2res_detector_shotnoise_model(--gain)
          Collapse the images with hdrl_imagelist_collapse(--collapse.*)
          Compute BPM from the collapsed master dark using
                  cr2res_bpm_compute(--bpm_kappa, --bpm_lines_ratio)
          Set the BPM in the master dark
          Compute the QCs with statistics and
                  cr2res_dark_qc_ron(--ron_hsize, --ron_nsamples)
        save master dark(g)
        save bpm(g)
    Library Functions used
      cr2res_detector_shotnoise_model()
      cr2res_bpm_compute()
      cr2res_bpm_from_mask()
      cr2res_dark_qc_ron()
      cr2res_bpm_count()
      cr2res_io_save_MASTER_DARK()
      cr2res_io_save_BPM()


OPTIONS
  The following options are provided by this recipe.

  --detector              : Only reduce the specified detector. [0]
  --bpm_method            : Method (DEFAULT, GLOBAL, LOCAL or RUNNING). [DEFAULT]
  --bpm_kappa             : Kappa Threshold for the BPM. [-1.0]
  --bpm_lines_ratio       : Maximum ratio of bad pixels per line. [0.5]
  --ron_hsize             : Half size of the window for RON computation. [6]
  --ron_nsamples          : Number of samples for RON computation. [100]
  --gain                  : Gain in [e- / ADU]. [2.1]
  --collapse.method       : Method used for collapsing the data. <MEAN |
                            WEIGHTED_MEAN | MEDIAN | SIGCLIP | MINMAX | MODE>
                            [MEAN]
  --collapse.sigclip.kappa-low  : Low kappa factor for kappa-sigma clipping
                                  algorithm. [3.0]
  --collapse.sigclip.kappa-high  : High kappa factor for kappa-sigma clipping
                                   algorithm. [3.0]
  --collapse.sigclip.niter  : Maximum number of clipping iterations for
                              kappa-sigma clipping. [5]
  --collapse.minmax.nlow: Low number of pixels to reject for the minmax clipping
                          algorithm. [1.0]
  --collapse.minmax.nhigh  : High number of pixels to reject for the minmax
                             clipping algorithm. [1.0]
  --collapse.mode.histo-min  : Minimum pixel value to accept for mode
                               computation. [10.0]
  --collapse.mode.histo-max  : Maximum pixel value to accept for mode
                               computation. [1.0]
  --collapse.mode.bin-size  : Binsize of the histogram. [0.0]
  --collapse.mode.method: Mode method (algorithm) to use. <MEDIAN | WEIGHTED |
                          FIT> [MEDIAN]
  --collapse.mode.error-niter  : Iterations to compute the mode error. [0]
```

10.2 cr2res_cal_detlin

```
NAME
  cr2res_cal_detlin -- Detector Linearity recipe

SYNOPSIS
```

```
  esorex [esorex-options] cr2res_cal_detlin [cr2res_cal_detlin-options] sof

DESCRIPTION
  Detector Linearity
    Measure the pixels non-linearity

    Inputs
      raw.fits DETLIN_DARK [3 to n]
      or raw.fits DETLIN_LAMP [3 to n]

    Outputs
      cr2res_cal_detlin_coeffs.fits CAL_DETLIN_COEFFS
      cr2res_cal_detlin_bpm.fits CAL_DETLIN_BPM

    Algorithm
      group the input raw frames by different settings
      loop on groups g:
        loop on detectors d:
          cr2res_cal_detlin_reduce() computes bpm(g, d) and coeffs(g, d)
          fill global_bpm(d) with bpm(g, d)
          fill global_coeffs(d) with coeffs(g, d)

        if (--single_settings)
          save bpm(g) file
          save coeffs(g) file
      save global_bpm file
      save global_coeffs file

      cr2res_cal_detlin_reduce()
        load input imlist and dits
        compute the traces (from 1. image, or collapsed if --trace_collapse)
          use cr2res_trace(--trace_smooth, --trace_degree,
                           --trace_min_cluster, --trace_opening)
        loop on the detector pixels pix:
          if the pixel is within a trace:
            cr2res_detlin_compute() computes polynomial(pix) and errors(pix)
        use the coeffs for the bpm computation
        set the bad pixel coefficients as NaN
        store the qc parameters in the returned property list

    Library Functions used
      cr2res_trace()
      cr2res_detlin_compute()
      cr2res_qc_detlin_gain()
      cr2res_qc_detlin_median()
      cr2res_qc_detlin_min_max_level()
      cr2res_io_save_BPM()
      cr2res_io_save_DETLIN_COEFFS()


OPTIONS
```

```
  The following options are provided by this recipe.

  --bpm_thresh           : BPM threshold, max % non-linear at 20000 ADU. [15.0]
  --trace_degree         : polynomial degree for the fit to the orders. [2]
  --trace_min_cluster    : size in pixels of the smallest allowed cluster. [5000]
  --trace_smooth_x       : Length of the smoothing kernel in x. [111]
  --trace_smooth_y       : Length of the smoothing kernel in y. [480]
  --trace_threshold      : Detection Threshold. [1.0]
  --trace_opening        : Use a morphological opening to rejoin clusters. [TRUE]
  --single_settings      : Create the products for each setting. [FALSE]
  --trace_collapse       : Collapse the input frames for the trace analysis.
                           [TRUE]
  --detector             : Only reduce the specified detector. [0]
  --plot_x               : X position for the plot. [0]
  --plot_y               : Y position for the plot. [0]
```

10.3 cr2res_cal_flat

```
NAME
  cr2res_cal_flat -- Flat recipe

SYNOPSIS
  esorex [esorex-options] cr2res_cal_flat [cr2res_cal_flat-options] sof

DESCRIPTION
  Flat
    Compute the master flat and perform the traces detection.
    If a TW file is provided, its traces and slit curvatures are used
    for the extraction needed to create the master flat.
    If not provided, the measured traces are used, and the slit is
    considered vertical.

    Inputs
      raw.fits FLAT [1 to n]
      trace.fits CAL_FLAT_TW [0 to 1]
              or CAL_FLAT_TW_MERGED
              or UTIL_TRACE_TW
              or UTIL_WAVE_TW
              or CAL_WAVE_TW
              or UTIL_SLIT_CURV_TW
      detlin.fits CAL_DETLIN_COEFFS [0 to 1]
      master_dark.fits CAL_DARK_MASTER [0 to 1]
      bpm.fits CAL_DARK_BPM [0 to 1]
            or CAL_FLAT_BPM
            or CAL_DETLIN_BPM
            or UTIL_BPM_SPLIT
            or UTIL_NORM_BPM
```

```
Outputs
  cr2res_cal_flat_[setting]_[Decker]_bpm.fits CAL_FLAT_BPM
  cr2res_cal_flat_[setting]_[Decker]_blaze.fits CAL_FLAT_EXTRACT_1D
  cr2res_cal_flat_[setting]_[Decker]_slit_model.fits CAL_FLAT_SLIT_MODEL
  cr2res_cal_flat_[setting]_[Decker]_slit_func.fits CAL_FLAT_SLIT_FUNC
  cr2res_cal_flat_[setting]_[Decker]_master_flat.fits CAL_FLAT_MASTER
  cr2res_cal_flat_[setting]_[Decker]_tw.fits CAL_FLAT_TW
  cr2res_cal_flat_[setting]_tw_merged.fits CAL_FLAT_TW_MERGED

Algorithm
  group the input frames by different settings
  loop on groups g:
    loop on decker positions p:
      loop on detectors d:
        cr2res_cal_flat_reduce() computes (master_flat, trace_wave,
               slit_func,extract_1d,slit_model,bpm)(g,p,d)
      Save slit_model(g,p)
      Save extract_1d(g,p)
      Save master_flat(g,p)
      Save trace_wave(g,p)
      Save slit_func(g,p)
      Save bpm(g,p)
    Merge the trace_wave(g,p,d) into trace_wave(g,d)
    Save trace_wave(g)

  cr2res_cal_flat_reduce()
    Load the images list
    Apply the detlin / dark / bpm calibrations
    Average the images to avg
    Compute the traces with cr2res_trace(--trace_degree,
               --trace_min_cluster, --trace_smooth, --trace_opening)
              on avg
    For the following step, the computed TW is used if there is no
    input TW provided, the provided one is used otherwise.
    loop on the traces t:
      cr2res_extract_slitdec_curved(--extract_oversample,
               --extract_swath_width, --extract_height,
               --extract_smooth_slit, --extract_smooth_spec)
        -> slit_func(t), extract_1d(t), slit_model(t)
    Compute the master flat with cr2res_master_flat(avg, slit_model,
               --bpm_low, --bpm_high, --bpm_lines_ratio)
      -> master_flat, bpm
    Merge the bpm with the input bpm
    Compute QCs
    store the qc parameters in the returned property list

Library functions used:
  cr2res_io_extract_decker_frameset()
  cr2res_trace()
  cr2res_extract_slitdec_curved()
  cr2res_master_flat()
```

```
      cr2res_qc_flat_lamp_ints()
      cr2res_qc_flat_mean_level()
      cr2res_qc_flat_mean_med_flux()
      cr2res_qc_flat_med_snr()
      cr2res_qc_overexposed()
      cr2res_qc_flat_trace_center_y()
      cr2res_trace_merge()
      cr2res_io_save_SLIT_MODEL()
      cr2res_io_save_EXTRACT_1D()
      cr2res_io_save_MASTER_FLAT()
      cr2res_io_save_TRACE_WAVE()
      cr2res_io_save_SLIT_FUNC()
      cr2res_io_save_BPM()


OPTIONS
  The following options are provided by this recipe.

  --subtract_nolight_rows  : Subtract the no-light rows. [FALSE]
  --cosmics               : Find and mark cosmic rays hits as bad. [FALSE]
  --bpm_low               : Low threshold for BPM detection. [0.8]
  --bpm_high              : High threshold for BPM detection. [1.2]
  --bpm_lines_ratio       : Maximum ratio of bad pixels per line. [0.5]
  --trace_degree          : polynomial degree for the fit to the orders. [2]
  --trace_min_cluster     : size in pixels of the smallest allowed cluster.
                            [100000]
  --trace_smooth_x        : Length of the smoothing kernel in x. [111]
  --trace_smooth_y        : Length of the smoothing kernel in y. [401]
  --trace_threshold       : Detection Threshold. [1.0]
  --trace_opening         : Use a morphological opening to rejoin clusters. [TRUE]
  --extract_method        : Extraction method (SUM / MEDIAN / TILTSUM / OPT_CURV
                            ). [OPT_CURV]
  --extract_oversample    : factor by which to oversample the extraction. [5]
  --extract_swath_width   : The swath width. [800]
  --extract_height        : Extraction height. [-1]
  --extract_smooth_slit   : Smoothing along the slit. [3.0]
  --extract_smooth_spec   : Smoothing along the spectrum. [2.0]
  --detector              : Only reduce the specified detector. [0]
  --order                 : Only reduce the specified order. [-1]
  --trace_nb              : Only reduce the specified trace number. [-1]
```

10.4 cr2res_cal_wave

```
NAME
  cr2res_cal_wave -- Wavelength Calibration

SYNOPSIS
  esorex [esorex-options] cr2res_cal_wave [cr2res_cal_wave-options] sof
```

```
DESCRIPTION
  Spectrum Extraction and Wavelength Calibration
    This recipe performs the extraction of the various orders along
    the provided traces, and the wavelength calibration of these
    extracted spectra.
    It can support different methods (--wl_method parameter):
      XCORR:  Cross Correlation with a emission lines catalog (default)
      LINE1D: Line identification and fitting for each 1D spectra
      LINE2D: Line identification and fitting for all 1D spectra at once

    Inputs
      raw.fits WAVE_UNE [1 to n]
            or WAVE_FPET [0 to n]
      trace.fits CAL_FLAT_TW [1]
              or CAL_FLAT_TW_MERGED
              or UTIL_TRACE_TW
              or UTIL_WAVE_TW
              or CAL_WAVE_TW
              or UTIL_SLIT_CURV_TW
      detlin.fits CAL_DETLIN_COEFFS [0 to 1]
      bpm.fits CAL_DARK_BPM [0 to 1]
            or CAL_FLAT_BPM
            or CAL_DETLIN_BPM
            or UTIL_BPM_MERGE
            or UTIL_BPM_SPLIT
      master_dark.fits CAL_DARK_MASTER [0 to 1]
      master_flat.fits CAL_FLAT_MASTER [0 to 1]
      lines.fits EMISSION_LINES [0 to 1]

    Outputs
      cr2res_cal_wave_tw.fits CAL_WAVE_TW
      cr2res_cal_wave_wave_map.fits CAL_WAVE_MAP
      cr2res_cal_wave_lines_diagnostics.fits CAL_WAVE_LINES_DIAGNOSTICS
      cr2res_cal_wave_extracted.fits CAL_WAVE_EXTRACT_1D

    Algorithm
      loop on detectors d:
        Call cr2res_cal_wave_reduce()
          -> out_trace_wave[une|fpet](d)
          -> lines_diagnostics[une|fpet](d)
          -> out_extracted[une|fpet](d)
          -> out_wave_map[une|fpet](d)
      Save out_trace_wave[une|fpet]
      Save lines_diagnostics[une|fpet]
      Save out_extracted[une|fpet]
      Save out_wave_map[une|fpet]

      cr2res_cal_wave_reduce():
        If FPET RAW frame is present, compute the slit_curvature from it,
        use it in the following, and store it in the out_trace_wave_fpet.
        Successively for UNE and FPET RAW frames:
          Load the raw image list
          Apply the calibrations to the image list
          Collapse the image list
          Extract along the traces from the collapsed image
          Compute the Wavelength with cr2res_wave_apply()
           -> out_trace_wave
           -> lines_diagnostics
           -> out_extracted
          Compute the Wavelength map
           -> out_wave_map

      cr2res_wave_apply()
        loop on the traces t:
          Get the spectrum
          Get the Initial guess
          Switch on the required method:
            CR2RES_LINE2D: cr2res_wave_2d()
            CR2RES_LINE1D: cr2res_wave_line_fitting()
            CR2RES_ETALON: cr2res_wave_etalon()
            CR2RES_XCORR:  cr2res_wave_xcorr()

    Library Functions used
      cr2res_io_find_TRACE_WAVE()
      cr2res_io_find_BPM()
      cr2res_io_read_dits()
      cr2res_io_load_image_list_from_set()
      cr2res_calib_imagelist()
      cr2res_io_load_TRACE_WAVE()
      cr2res_extract_traces()
      cr2res_wave_apply()
      cr2res_extract_EXTRACT1D_get_spectrum()
      cr2res_wave_estimate_compute()
      cr2res_wave_2d()
      cr2res_wave_line_fitting()
      cr2res_wave_etalon()
      cr2res_wave_xcorr()
      cr2res_wave_gen_wave_map()
      cr2res_io_save_TRACE_WAVE()
      cr2res_io_save_WAVE_MAP()
      cr2res_io_save_LINES_DIAGNOSTICS()
      cr2res_io_save_EXTRACT_1D()


OPTIONS
  The following options are provided by this recipe.

  --detector             : Only reduce the specified detector. [0]
  --order                : Only reduce the specified order. [-1]
  --trace_nb             : Only reduce the specified trace number. [-1]
  --slit_degree          : Slit fitting Polynomial degree (1 or 2). [2]
  --subtract_nolight_rows  : Subtract the no-light rows. [FALSE]
  --cosmics              : Find and mark cosmic rays hits as bad. [FALSE]
  --collapse_method      : Collapse the input images (MEAN or MEDIAN). [MEDIAN]
  --extract_oversample   : factor by which to oversample the extraction. [7]
  --extract_swath_width  : The swath width. [2048]
  --extract_height       : Extraction height. [160]
  --extract_smooth_slit  : Smoothing along the slit. [15.0]
  --extract_smooth_spec  : Smoothing along the spectrum. [2e-07]
  --wl_method            : Wavelength Method (XCORR / LINE1D / LINE2D). [XCORR]
  --wl_shift             : Wavelength shift (nm) to apply to the guess. [0.0]
  --wl_est               : Estimated wavelength start and end. [-1.0, -1.0]
  --wl_err               : Estimated wavelength error (in nm). [0.04]
  --wl_degree            : Wavelength Polynomial degree. [0]
  --log                  : Flag for taking the Log() value of the lines. [FALSE]
  --fallback             : Flag for using the input WL when no computation.
                           [FALSE]
  --keep                 : Flag for re-using higher degrees of first guess in
                           Cross-Corr. [TRUE]
  --save                 : Flag to save UNE results (if UNE and FPET are used as
                           inputs). [FALSE]
  --clean_spectrum       : Flag to automatically clean the missing lines. [TRUE]
  --display              : Flag for display. [FALSE]
  --display_range        : Wavelength range to display [start, end] (in nm).
                           [-1.0, -1.0]
```

10.5 cr2res_obs_2d

```
NAME
  cr2res_obs_2d -- 2D Observation recipe

SYNOPSIS
  esorex [esorex-options] cr2res_obs_2d [cr2res_obs_2d-options] sof

DESCRIPTION
  2D Observation
    This recipe is meant for extended objects. In each trace, the pixels
    are calibrated, and stored in the output table.

    Inputs
      raw.fits OBS_2D_OBJECT [1 to n]
           and OBS_2D_SKY [0 to n]
      trace.fits CAL_FLAT_TW [1]
              or CAL_FLAT_TW_MERGED
              or UTIL_TRACE_TW
              or UTIL_WAVE_TW
              or CAL_WAVE_TW
              or UTIL_SLIT_CURV_TW
      detlin.fits CAL_DETLIN_COEFFS [0 to 1]
      bpm.fits CAL_DARK_BPM [0 to 1]
            or CAL_FLAT_BPM
            or CAL_DETLIN_BPM
            or UTIL_BPM_MERGE
            or UTIL_BPM_SPLIT
      master_dark.fits CAL_DARK_MASTER [0 to 1]
      master_flat.fits CAL_FLAT_MASTER [0 to 1]

    Outputs
      cr2res_obs_2d_frame_<nb>_extracted.fits OBS_2D_EXTRACT
      cr2res_obs_2d_frame_<nb>_calibrated.fits OBS_2D_CALIBRATED

    Algorithm
      loop on raw frames f:
        loop on detectors d:
          cr2res_obs_2d_reduce()
            -> extract(d)
        Save extract

      cr2res_obs_2d_reduce()
        Load the input image
        Apply the calibrations to the image
        Load the trace wave table
        Call cr2res_extract2d_traces()
          -> extract

    Library Functions used
      cr2res_io_find_TRACE_WAVE()
      cr2res_io_find_BPM()
      cr2res_obs_2d_reduce()
      cr2res_obs_2d_check_inputs_validity()
      cr2res_pfits_get_dit()
      cr2res_io_load_image()
      cr2res_calib_image()
      cr2res_io_load_TRACE_WAVE()
      cr2res_extract2d_traces()
      cr2res_io_save_EXTRACT_2D()
      cr2res_io_save_CALIBRATED()


OPTIONS
  The following options are provided by this recipe.

  --subtract_nolight_rows   : Subtract median row from baffled region at detector
                              bottom. [FALSE]
  --subtract_interorder_column   : Subtract column-by-column fit to the pixel
                                   values between spectral orders. [TRUE]
  --cosmics                : Find and mark cosmic rays hits as bad. [FALSE]
  --detector               : Only reduce the specified detector. [0]
  --order                  : Only reduce the specified order. [-1]
  --trace_nb            : Only reduce the specified trace number. [-1]
```

10.6 cr2res_obs_nodding

```
NAME
  cr2res_obs_nodding -- Nodding Observation recipe

SYNOPSIS
  esorex [esorex-options] cr2res_obs_nodding [cr2res_obs_nodding-options] sof

DESCRIPTION
  Nodding Observation
    This recipe handles nodding observations. It expects an even number
    of rawframes in input, and as many A positions as B positions
    If the input are standard stars, it computes a post processing step
    to determine the throughput.
    If the input are spectro astrometric data, it will apply the nodding
    on each of the sub-groups

    Inputs
      raw.fits CAL_NODDING_OTHER [2 to 2n]
            or CAL_NODDING_JITTER [2 to 2n]
            or OBS_NODDING_OTHER [2 to 2n]
            or OBS_NODDING_JITTER [2 to 2n]
            or OBS_ASTROMETRY_OTHER [2 to 2n]
            or OBS_ASTROMETRY_JITTER [2 to 2n]
      trace.fits CAL_FLAT_TW [1]
              or CAL_FLAT_TW_MERGED
              or UTIL_TRACE_TW
              or UTIL_WAVE_TW
              or CAL_WAVE_TW
              or UTIL_SLIT_CURV_TW
      detlin.fits CAL_DETLIN_COEFFS [0 to 1]
      bpm.fits CAL_DARK_BPM [0 to 1]
            or CAL_FLAT_BPM
            or CAL_DETLIN_BPM
            or UTIL_BPM_MERGE
            or UTIL_BPM_SPLIT
      master_dark.fits CAL_DARK_MASTER [0 to 1]
      master_flat.fits CAL_FLAT_MASTER [0 to 1]
      photo_flux.fits PHOTO_FLUX [0 to 1]
      blaze.fits CAL_FLAT_EXTRACT_1D [0 to 1]

    Outputs
      cr2res_obs_nodding_extractedA.fits OBS_NODDING_EXTRACTA
      cr2res_obs_nodding_extractedB.fits OBS_NODDING_EXTRACTB
      cr2res_obs_nodding_combinedA.fits OBS_NODDING_COMBINEDA
      cr2res_obs_nodding_combinedB.fits OBS_NODDING_COMBINEDB
      cr2res_obs_nodding_trace_wave_A.fits OBS_NODDING_TWA
      cr2res_obs_nodding_trace_wave_B.fits OBS_NODDING_TWB
      cr2res_obs_nodding_modelA.fits OBS_NODDING_SLITMODELA
      cr2res_obs_nodding_modelB.fits OBS_NODDING_SLITMODELB
      cr2res_obs_nodding_slitfuncA.fits OBS_NODDING_SLITFUNCA
      cr2res_obs_nodding_slitfuncB.fits OBS_NODDING_SLITFUNCB
      cr2res_obs_nodding_throughput.fits OBS_NODDING_THROUGHPUT

    Algorithm
      loop on detectors d:
        call cr2res_obs_nodding_reduce()
          -> combined[a|b](d)
          -> extract[a|b|c](d)
          -> slitfunc[a|b](d)
          -> model[a|b](d)
          -> tw[a|b](d)
      Save combineda and combinedb
      Save extracta, extractb, extractc
      Save slitfunca and slitfuncb
      Save modela and modelb
      Save throughput

      cr2res_obs_nodding_reduce()
        Load the input raw frames in an image list
        Apply the calibrations to the image list
        Split the list in listA and listB image lists
        Compute diffA=listA-listB and diffB=listB-listA
        Collapse diffA and diffB
          -> combined[a|b]
        Load the input trace wave
        Compute the slit fractions for A and B by using the nodthrow
                and the assumption that A and B are at equal distances
                from the slit center.
        Compute 2 new trace_wave files  with these 2 computed slit fractions
        Extract the spectra for A and for B
          -> extracted[a|b]
          -> slit_func[a|b]
          -> model_master[a|b]
          -> trace_wave[a|b]
        Compute QC parameters
        If STD star, compute the throughput
          -> throughput

    Library functions used
      cr2res_io_find_TRACE_WAVE()
      cr2res_io_find_BPM()
      cr2res_obs_nodding_reduce()
      cr2res_nodding_read_positions()
      cr2res_io_read_dits()
      cr2res_io_load_image_list_from_set()
      cr2res_calib_imagelist()
```

```
      cr2res_combine_nodding_split()
      cr2res_io_load_TRACE_WAVE()
      cr2res_pfits_get_nodthrow()
      cr2res_trace_new_slit_fraction()
      cr2res_extract_traces()
      cr2res_qc_obs_nodding_signal()
      cr2res_qc_obs_nodding_transmission()
      cr2res_qc_obs_slit_psf()
      cr2res_photom_engine()
      cr2res_io_save_COMBINED()
      cr2res_io_save_EXTRACT_1D()
      cr2res_io_save_SLIT_FUNC()
      cr2res_io_save_SLIT_MODEL()
      cr2res_io_save_THROUGHPUT()


OPTIONS
  The following options are provided by this recipe.

  --subtract_nolight_rows  : Subtract median row from baffled region at detector
                             bottom. [FALSE]
  --subtract_interorder_column  : Subtract column-by-column fit to the pixel
                                  values between orders. [TRUE]
  --cosmics               : Find and mark cosmic rays hits as bad. [FALSE]
  --error_method          : The 1d extraction error calculation method. <Poisson |
                            Horne> [Poisson]
  --nodding_invert        : Flag to use when A is above B. [FALSE]
  --extract_oversample    : factor by which to oversample the extraction. [7]
  --extract_swath_width   : The swath width. [800]
  --extract_height        : Extraction height. [-1]
  --extract_smooth_slit   : Smoothing along the slit. [1.0]
  --extract_smooth_spec   : Smoothing along spectrum. [8e-08]
  --detector              : Only reduce the specified detector. [0]
  --idp                   : Flag to produce  IDP files. [FALSE]
  --display_detector      : Apply the display for the specified detector. [0]
  --display_order         : Apply the display for the specified order. [1000]
  --display_trace         : Apply the display for the specified trace. [0]
```

10.7 cr2res_obs_pol

```
NAME
  cr2res_obs_pol -- Polarimetry Observation recipe

SYNOPSIS
  esorex [esorex-options] cr2res_obs_pol [cr2res_obs_pol-options] sof

DESCRIPTION
  Polarimetry Observation
```

```
The input raw frames are separated in A and B nodding positions. A
and B frames are reduced separately. The frames are grouped by blocks
of 4 frames. Each block generates 8 extractions. For each order
found, 8 spectra are passed to the demodulation functions.
The results are stored in polarimetry tables (1 per group of 4
frames). The polarimetry tables are then merged together if there are
several group of 4 frames available.

Inputs
  raw.fits OBS_POLARIMETRY_OTHER [4 to 4n]
  trace.fits CAL_FLAT_TW [1]
          or CAL_FLAT_TW_MERGED
          or UTIL_TRACE_TW
          or UTIL_WAVE_TW
          or CAL_WAVE_TW
          or UTIL_SLIT_CURV_TW
  detlin.fits CAL_DETLIN_COEFFS [0 to 1]
  bpm.fits CAL_DARK_BPM [0 to 1]
        or CAL_FLAT_BPM
        or CAL_DETLIN_BPM
        or UTIL_BPM_MERGE
        or UTIL_BPM_SPLIT
  master_dark.fits CAL_DARK_MASTER [0 to 1]
  master_flat.fits CAL_FLAT_MASTER [0 to 1]
  blaze.fits CAL_FLAT_EXTRACT_1D [0 to 1]

Outputs
  cr2res_obs_pol_specA.fits OBS_POL_SPECA
  cr2res_obs_pol_specB.fits OBS_POL_SPECB

Algorithm
  loop on detectors d:
    cr2res_obs_pol_reduce()
      -> pol_speca(d)
      -> pol_specb(d)
  Save pol_speca
  Save pol_specb

  cr2res_obs_pol_reduce():
    Read the nodding positions of the raw frames
    Split the rawframes in A and B positions
    Call cr2res_obs_pol_reduce_one() successively on A and B
      -> pol_speca
      -> pol_specb

  cr2res_obs_pol_reduce_one():
    Read the DITs
    Read the decker positions
    Load the image list
    Apply the calibrations to the image list
    Load the trace_wave
```

```
        For each group of 4 images g:
          Compute 8 extracted tables (2 per image):
              1u, 1d, 2u, 2d, 3u, 3d, 4u, 4d
              where u/d are for up and down
                    1->4 is derived with cr2res_pol_sort_frames()
                    The decker info is used to derive the 8 slit fractions
          Count norders the number of different orders in those 8 tables
              [Note : 1 extracted table has 1 spectrum per order]
          loop on orders o:
            Get the 8 spectra/wl/error for this order from
                  1u, 1d, 2u, 2d, 3u, 3d, 4u, 4d
            Call the cr2res_pol_demod_stokes()
              -> demod_stokes(o, g)
            Call cr2res_pol_demod_null()
              -> demod_null(o, g)
            Call cr2res_pol_demod_intens()
              -> demod_intens(o, g)
          Create pol_spec(g) from demod_stokes(g),
                                  demod_null(g),
                                  demod_intens(g)
        Merge pol_spec(g) into pol_spec

    Library functions used
      cr2res_io_find_TRACE_WAVE()
      cr2res_io_find_BPM()
      cr2res_obs_pol_reduce()
      cr2res_nodding_read_positions()
      cr2res_combine_nodding_split_frames()
      cr2res_obs_pol_reduce_one()
      cr2res_io_read_dits()
      cr2res_io_read_decker_positions()
      cr2res_io_load_image_list_from_set()
      cr2res_calib_imagelist()
      cr2res_io_load_TRACE_WAVE()
      cr2res_pol_sort_frames()
      cr2res_trace_slit_fraction_create()
      cr2res_trace_new_slit_fraction()
      cr2res_extract_traces()
      cr2res_obs_pol_get_order_numbers()
      cr2res_pol_demod_stokes()
      cr2res_pol_demod_null()
      cr2res_pol_demod_intens()
      cr2res_pol_POL_SPEC_create()
      cr2res_pol_spec_pol_merge()
      cr2res_io_save_POL_SPEC()


OPTIONS
  The following options are provided by this recipe.

  --subtract_nolight_rows  : Subtract median row from baffled region at detector
```

```
                                  bottom. [FALSE]
  --subtract_interorder_column   : Subtract column-by-column fit to the pixel
                                   values between orders. [TRUE]
  --cosmics               : Find and mark cosmic rays hits as bad. [FALSE]
  --extract_oversample    : factor by which to oversample the extraction. [5]
  --extract_swath_width : The swath width. [800]
  --extract_height        : Extraction height. [-1]
  --extract_smooth_spec : Smoothing along the spectrum. [0.0]
  --extract_smooth_slit : Smoothing along the slit. [2.0]
  --detector              : Only reduce the specified detector. [0]
  --idp                   : Flag to produce  IDP files. [FALSE]
  --save_group            : Save extra files for the specified group number (0: no
                            save). [0]
```

10.8 cr2res_obs_staring

```
NAME
  cr2res_obs_staring -- Staring Observation recipe

SYNOPSIS
  esorex [esorex-options] cr2res_obs_staring [cr2res_obs_staring-options] sof

DESCRIPTION
  Staring Observation
    This recipe handles staring observations.

    Inputs
      raw.fits OBS_STARING_OTHER [1 to n]
            or OBS_STARING_JITTER [1 to n]
            or OBS_WAVE_SKY [1 to n]
      trace.fits CAL_FLAT_TW [1]
              or CAL_FLAT_TW_MERGED
              or UTIL_TRACE_TW
              or UTIL_WAVE_TW
              or CAL_WAVE_TW
              or UTIL_SLIT_CURV_TW
      detlin.fits CAL_DETLIN_COEFFS [0 to 1]
      bpm.fits CAL_DARK_BPM [0 to 1]
            or CAL_FLAT_BPM
            or CAL_DETLIN_BPM
            or UTIL_BPM_MERGE
            or UTIL_BPM_SPLIT
      master_dark.fits CAL_DARK_MASTER [0 to 1]
      master_flat.fits CAL_FLAT_MASTER [0 to 1]
      blaze.fits CAL_FLAT_EXTRACT_1D [0 to 1]

    Outputs
   cr2res_obs_staring_slitfunc.fits OBS_STARING_SLITFUNC
   cr2res_obs_staring_model.fits OBS_STARING_SLITMODEL
   cr2res_obs_staring_extracted.fits OBS_STARING_EXTRACT

    Algorithm
      loop on detectors d:
        call cr2res_obs_staring_reduce()
          -> combined(d)
          -> extract(d)
          -> slitfunc(d)
          -> model(d)
      Save extract
      Save slitfunc
      Save model

      cr2res_obs_staring_reduce()
        Load the input raw frames in an image list
        Apply the calibrations to the image list
        Collapse the image list
        Load the input trace wave
        Recompute a new trace wave with the specified slit fraction
               (--slit_frac) if needed
        Extract the spectra from the collapsed image
          -> combined
          -> extracted
          -> slit_func
          -> model_master
        Compute QC parameters

    Library functions used
      cr2res_io_find_TRACE_WAVE()
      cr2res_io_find_BPM()
      cr2res_obs_staring_reduce()
      cr2res_io_read_dits()
      cr2res_io_load_image_list_from_set()
      cr2res_calib_imagelist()
      cr2res_io_load_TRACE_WAVE()
      cr2res_extract_traces()
      cr2res_io_save_COMBINED()
      cr2res_io_save_EXTRACT_1D()
      cr2res_io_save_SLIT_FUNC()


OPTIONS
  The following options are provided by this recipe.

  --subtract_nolight_rows   : Subtract median row from baffled region at detector
                              bottom. [FALSE]
  --subtract_interorder_column   : Subtract column-by-column fit to the pixel
                                   values between spectral orders. [TRUE]
  --cosmics                : Find and mark cosmic rays hits as bad. [FALSE]
  --slit_frac              : Wished slit fraction. [-1.0, -1.0]
  --extract_oversample  : factor by which to oversample the extraction. [5]
  --extract_swath_width : The swath width. [800]
  --extract_height      : Extraction height. [-1]
  --extract_smooth_slit : Smoothing along the slit (1 for high S/N, 5 for low).
                          [2.0]
  --extract_smooth_spec : Smoothing along the spectrum. [0.0]
  --detector            : Only reduce the specified detector. [0]
  --idp                 : Flag to produce  IDP files. [FALSE]
  --display_order       : Apply the display for the specified order. [0]
  --display_trace       : Apply the display for the specified trace. [0]
```

10.9 cr2res_util_bpm_merge

```
NAME
  cr2res_util_bpm_merge -- BPM merging utility

SYNOPSIS
  esorex [esorex-options] cr2res_util_bpm_merge [cr2res_util_bpm_merge-options] sof

DESCRIPTION
  BPM merging
    Each input BPM is split into several BPMs

    Inputs
      raw.fits CAL_DARK_BPM [1 to n]
            or CAL_FLAT_BPM
            or CAL_DETLIN_BPM
            or UTIL_BPM_SPLIT
            or UTIL_BPM_MERGE
            or UTIL_NORM_BPM

    Outputs
      <recipe_name>.fits UTIL_BPM_MERGE

    Algorithm
          read input BPMs and merge their types by adding the
          integers, in powers of 2.

    Library functions used:
      cr2res_io_load_BPM()
      cr2res_io_save_BPM()


OPTIONS
  A single option is provided by this recipe.

  --detector            : Only reduce the specified detector. [0]
```

10.10 cr2res_util_bpm_split

```
NAME
  cr2res_util_bpm_split -- BPM splitting utility

SYNOPSIS
  esorex [esorex-options] cr2res_util_bpm_split [cr2res_util_bpm_split-options] sof

DESCRIPTION
  BPM splitting
    Each input BPM is split into several BPMs

    Inputs
      raw.fits CAL_DARK_BPM [1 to n]
            or CAL_FLAT_BPM
            or CAL_DETLIN_BPM
            or UTIL_BPM_MERGE
            or UTIL_BPM_SPLIT
            or UTIL_NORM_BPM

    Outputs
      <input_name>_split_<bpm_code>.fits UTIL_BPM_SPLIT

    Algorithm
      loop on input raw frames f:
        loop on detectors d:
          loop on bpm types t:
            call cr2res_bpm_from_mask()
             -> bpm_single_type(t, d, f)
        loop on bpm types t:
          Save bpm_single_type(f, t) (UTIL_BPM_SPLIT)

    Library functions used:
      cr2res_io_load_BPM()
      cr2res_bpm_from_mask()
      cr2res_io_save_BPM()


OPTIONS
  A single option is provided by this recipe.

  --detector            : Only reduce the specified detector. [0]
```

10.11 cr2res_util_calib

```
NAME
  cr2res_util_calib -- Calibration utility
```

```
SYNOPSIS
  esorex [esorex-options] cr2res_util_calib [cr2res_util_calib-options] sof

DESCRIPTION
  Frames Calibration
    Each input file is corrected with BPM / Dark / Flat / Det.Lin. / Cosmics

    Inputs
      raw.fits FLAT [1 to n]
            or WAVE_UNE
            or WAVE_FPET
            or CAL_NODDING_OTHER
            or CAL_NODDING_JITTER
            or OBS_NODDING_OTHER
            or OBS_NODDING_JITTER
            or OBS_ASTROMETRY_OTHER
            or OBS_ASTROMETRY_JITTER
            or OBS_STARING_OTHER
            or OBS_STARING_JITTER
            or OBS_WAVE_SKY
            or OBS_POLARIMETRY_OTHER
            or OBS_2D_OBJECT
            or OBS_2D_SKY
      detlin.fits CAL_DETLIN_COEFFS [0 to 1]
      bpm.fits CAL_DARK_BPM [0 to 1]
            or CAL_FLAT_BPM
            or CAL_DETLIN_BPM
            or UTIL_BPM_MERGE
            or UTIL_BPM_SPLIT
      master_dark.fits CAL_DARK_MASTER [0 to 1]
      master_flat.fits CAL_FLAT_MASTER [0 to 1]

    Outputs
      <input_name>_calibrated.fits UTIL_CALIB
      or
      <input_name>_calibrated_collapsed.fits UTIL_CALIB

    Algorithm
      loop on detectors d:
        Call cr2res_calib_imagelist()() to calibrate cosmics,
                 detlin, bpm, dark, flat
          -> calibrated(d)
        Collapse the calibrated image list to collapsed(d)
        if (collapse) save collapsed
        else save every individual calibrated frame
    Library functions used:
      cr2res_io_load_image()
      cr2res_calib_imagelist()
      cr2res_io_save_CALIBRATED()
```

```
OPTIONS
  The following options are provided by this recipe.

  --detector             : Only reduce the specified detector. [0]
  --clean_bad            : Apply the cleaning to the bad pixels. [FALSE]
  --subtract_nolight_rows   : Subtract median row from baffled region at detector
                              bottom. [FALSE]
  --subtract_interorder_column   : Subtract column-by-column fit to the pixel
                                   values between spectral orders. [TRUE]
  --cosmics              : Find and mark cosmic rays hits as bad. [FALSE]
  --collapse             : Collapse the input (NONE, SUM, MEAN or MEDIAN). [NONE]
```

10.12 cr2res_util_extract

```
NAME
  cr2res_util_extract -- Optimal Extraction utility

SYNOPSIS
  esorex [esorex-options] cr2res_util_extract [cr2res_util_extract-options] sof

DESCRIPTION
  Spectrum Extraction
    This utility performs the optimal extraction along precomputed traces

    Inputs
      raw.fits FLAT [1 to n]
            or UTIL_CALIB
            or WAVE_UNE
            or WAVE_FPET
            or CAL_NODDING_OTHER
            or CAL_NODDING_JITTER
            or OBS_NODDING_OTHER
            or OBS_NODDING_JITTER
            or OBS_ASTROMETRY_OTHER
            or OBS_ASTROMETRY_JITTER
            or OBS_STARING_OTHER
            or OBS_STARING_JITTER
            or OBS_WAVE_SKY
            or OBS_POLARIMETRY_OTHER
            or OBS_2D_OBJECT
            or OBS_2D_SKY
      trace.fits CAL_FLAT_TW [1]
              or CAL_FLAT_TW_MERGED
              or UTIL_TRACE_TW
              or UTIL_WAVE_TW
              or CAL_WAVE_TW
              or UTIL_SLIT_CURV_TW
      slitfunc.fits CAL_FLAT_SLIT_FUNC [0 to 1]
                 or UTIL_SLIT_FUNC
                 or OBS_NODDING_SLITFUNCA
                 or OBS_NODDING_SLITFUNCB
                 or OBS_STARING_SLITFUNC
      bpm.fits CAL_DARK_BPM [0 to 1]
            or CAL_FLAT_BPM
            or CAL_DETLIN_BPM
            or UTIL_BPM_MERGE
            or UTIL_BPM_SPLIT

    Outputs
      <input_name>_extr1D.fits UTIL_EXTRACT_1D
      <input_name>_extrSlitFu.fits UTIL_SLIT_FUNC
      <input_name>_extrModel.fits UTIL_SLIT_MODEL

    Algorithm
      loop on raw frames f:
        loop on detectors d:
          Load the trace wave
          Recompute a new trace wave with the specified slit fraction
                   (--slit_frac) if needed
          Load the image to extract
          Load the BPM and set them in the image
          Load the input slit_func if available
          Run the extraction cr2res_extract_traces(--method,--height,
                   --swath_width,--oversample,--smooth_slit,--smooth_spec)
            -> creates SLIT_MODEL(f,d), SLIT_FUNC(f,d), EXTRACT_1D(f,d)
        Save SLIT_MODEL(f), SLIT_FUNC(f), EXTRACT_1D(f)

    Library functions used
      cr2res_io_load_TRACE_WAVE()
      cr2res_trace_new_slit_fraction()
      cr2res_io_load_image()
      cr2res_io_load_BPM()
      cr2res_extract_traces()
      cr2res_io_save_SLIT_MODEL()
      cr2res_io_save_SLIT_FUNC()
      cr2res_io_save_EXTRACT_1D()


OPTIONS
  The following options are provided by this recipe.

  --oversample          : factor by which to oversample the extraction. [5]
  --swath_width         : The swath width. [800]
  --height              : Extraction height. [-1]
  --smooth_slit         : Smoothing along the slit at extraction. [2.0]
  --smooth_spec         : Smoothing spectrum at extraction. [0.0]
  --method              : Extraction method (SUM / MEDIAN / TILTSUM / OPT_CURV
                          ). [OPT_CURV]
  --slit_frac           : Wished slit fraction. [-1.0, -1.0]
  --detector            : Only reduce the specified detector. [0]
  --order               : Only reduce the specified order. [-1]
  --trace_nb            : Only reduce the specified trace number. [-1]
```

10.13 cr2res_util_genlines

```
NAME
  cr2res_util_genlines -- Generate spectrum calibration FITS tables

SYNOPSIS
  esorex [esorex-options] cr2res_util_genlines [cr2res_util_genlines-options] sof

DESCRIPTION
   Generate Lines calibration tables

    Inputs
      raw1.txt EMISSION_LINES_TXT [1]
      raw2.txt LINES_SELECTION_TXT [0 to 1]
      The ASCII file raw1.txt must contain two columns:
        1st: Wavelengths in increasing order (the unit is corrected by
                  the factor option to obtain nanometers).
        2nd: The atmospheric emission.
        The file is in the catalogs/ directory of the CR2RES distribution.
      The optional ASCII files raw2.txt contain a list of wavelength
        ranges (1 per line) of type: 1632.25,1632.70
        The files are in the catalogs/selection/ directory of the CR2RES
            distribution. They are called XXX.dat, where XXX
            identifies the setting [L3244, L3340, ...].

    Output
      cr2res_util_genlines.fits EMISSION_LINES
      cr2res_util_genlines_XXX.fits EMISSION_LINES

    Algorithm
      Parse and load the 2 columns raw1.txt file
      Apply the --wl_factor correction
      if (--display) plot it
      Create the CPL table
      Save the table with all lines
      Loop on the selection files
          Only keep the lines that fall in the selection ranges (if any)
          Save the table with the selected lines

    Library functions used
      cr2res_io_save_EMISSION_LINES()
```

```
OPTIONS
  The following options are provided by this recipe.

  --wl_factor           : The factor used to multiply the wl. [1.0]
  --display             : Flag to plot. [FALSE]
```

10.14 cr2res_util_genstd

```
NAME
  cr2res_util_genstd -- Generate standard star FITS tables

SYNOPSIS
  esorex [esorex-options] cr2res_util_genstd [cr2res_util_genstd-options] sof

DESCRIPTION
  Photospheric flux table generation

    This utility is used to generate the photospheric flux table

    Inputs
      raw.txt PHOTO_FLUX_TXT [1 to n]
      The specified files are named after the standard star they represent
      (e.g. HIP61007.txt).
      The first line of the file must contain the RA and DEC (hh mm ss).
      (e.g. # 13 20 35.818        -36 42 44.26).
      The rest of the file must contain two columns:
      1st: Wavelengths in increasing order
      2nd: The atmospheric emission

    Outputs
      cr2res_util_genstd.fits PHOTO_FLUX

    Algorithm

    Library functions used


OPTIONS
  A single option is provided by this recipe.

  --display             : Flag to plot. [FALSE]
```

10.15 cr2res_util_normflat

```
NAME
```

```
  cr2res_util_normflat -- Flat Normalization utility

SYNOPSIS
  esorex [esorex-options] cr2res_util_normflat [cr2res_util_normflat-options] sof

DESCRIPTION
  Flat normalization
    The input RAW files are grouped by setting/decker and each group is
    reduced separately. For each group, a slit model file with matching
    setting/decker is expected.

    Inputs
      raw.fits FLAT [1 to n]
            or UTIL_CALIB
      slit_model.fits CAL_FLAT_SLIT_MODEL [1 to m]
                   or UTIL_SLIT_MODEL
                   or OBS_NODDING_SLITMODELA
                   or OBS_NODDING_SLITMODELB

    Outputs
      cr2res_util_normflat_[setting]_[Decker]_bpm.fits UTIL_NORM_BPM
      cr2res_util_normflat_[setting]_[Decker]_master.fits UTIL_MASTER_FLAT

    Algorithm
      group the input frames by different settings
      loop on groups g:
        group the input frames by different decker positions
        loop on decker positions p:
          loop on detectors d:
            cr2res_util_normflat_reduce() computes (master_flat,bpm)(g,p,d)
        Save master_flat(g,p)
        Save bpm(g,p)

      cr2res_util_normflat_reduce()
        Load the images list
        Average the images to avg
        Load the input slit_model with the proper setting/decker
        Compute the master flat with cr2res_master_flat(avg,
                 slit_model, --bpm_low, --bpm_high, --bpm_lines_ratio)
          -> master_flat, bpm

  Library functions used:
      cr2res_extract_frameset()
      cr2res_io_extract_decker_frameset()
      cr2res_io_find_SLIT_MODEL()
      cr2res_io_load_image_list_from_set()
      cr2res_io_load_SLIT_MODEL()
      cr2res_master_flat()
      cr2res_io_save_MASTER_FLAT()
      cr2res_io_save_BPM()
```

```
OPTIONS
  The following options are provided by this recipe.

  --bpm_low              : Low threshold for BPM detection. [0.5]
  --bpm_high             : High threshold for BPM detection. [2.0]
  --bpm_lines_ratio      : Maximum ratio of bad pixels per line. [0.5]
  --detector             : Only reduce the specified detector. [0]
```

10.16 cr2res_util_plot

```
NAME
  cr2res_util_plot -- Plotting utility

SYNOPSIS
  esorex [esorex-options] cr2res_util_plot [cr2res_util_plot-options] sof

DESCRIPTION
  Plotting
    This utility detects the type of the first passed file, and plots
    relevant graphs accordingly.

    Inputs
      first.fits CATALOG [1]
              or EXTRACT_1D
      second.fits CATALOG [0 to 1]
    Outputs
      -

    Algorithm

    Library Functions used


OPTIONS
  The following options are provided by this recipe.

  --xmin                 : Minimum x value to plot. [-1.0]
  --xmax                 : Maximum x value to plot. [-1.0]
  --detector             : Only reduce the specified detector. [0]
  --order                : Only reduce the specified order. [-1]
  --trace_nb             : Only reduce the specified trace number. [-1]
  --adjust_level         : Flag to adjust the level with 2 plots. [TRUE]
```

10.17 cr2res_util_slit_curv

```
NAME
  cr2res_util_slit_curv -- Slit Curvature utility

SYNOPSIS
  esorex [esorex-options] cr2res_util_slit_curv [cr2res_util_slit_curv-options] sof

DESCRIPTION
  Slit curvature computation
    For each input trace_wave file, the slit curvature is derived.

    Inputs
      trace.fits CAL_FLAT_TW [1 to n]
              or CAL_FLAT_TW_MERGED
              or UTIL_TRACE_TW
              or UTIL_WAVE_TW
              or CAL_WAVE_TW
              or UTIL_SLIT_CURV_TW
      lamp.fits WAVE_FPET [1 to n]

    Outputs
      <input_lamp_name>_map.fits UTIL_SLIT_CURV_MAP
      <input_lamp_name>_tw.fits UTIL_SLIT_CURV_TW

    Algorithm
      loop on input raw files pairs (t,l):
        loop on detectors d:
          Load the TRACE_WAVE table for the current detector
          Loop over the traces t:
            Call cr2res_slit_curv_compute_order_trace() to get the
                  current trace curvatures
            Update the slit curvature in the TRACE_WAVE table
          Generate a slit curve map
        Save the TRACE_WAVE
        Save the SLIT_CURVE_MAP

    Library functions used
      cr2res_io_load_TRACE_WAVE()
      cr2res_slit_curv_compute_order_trace()
      cr2res_slit_curv_gen_map()
      cr2res_io_save_SLIT_CURV_MAP()
      cr2res_io_save_TRACE_WAVE()


OPTIONS
  The following options are provided by this recipe.

  --detector              : Only reduce the specified detector. [0]
  --order                 : Only reduce the specified order. [-1]
  --trace_nb             : Only reduce the specified trace number. [-1]
  --degree               : Polynomial degree for the fit (1 or 2). [2]
  --display              : X value to display (1->2048). [0]
```

10.18 cr2res_util_splice

```
NAME
  cr2res_util_splice -- Splicing utility

SYNOPSIS
  esorex [esorex-options] cr2res_util_splice [cr2res_util_splice-options] sof

DESCRIPTION
  Continuum normalization and splicing together spectra.


    Inputs
      blaze.fits CAL_FLAT_EXTRACT_1D [1]
      trace.fits CAL_FLAT_TW [1]
              or CAL_FLAT_TW_MERGED
              or UTIL_TRACE_TW
              or UTIL_WAVE_TW
              or CAL_WAVE_TW
              or UTIL_SLIT_CURV_TW
      extracted.fits UTIL_EXTRACT_1D [1]

    Outputs

    Algorithm

    Library functions used:


OPTIONS
  A single option is provided by this recipe.

  --detector             : Only reduce the specified detector. [0]
```

10.19 cr2res_util_trace

```
NAME
  cr2res_util_trace -- Trace utility

SYNOPSIS
  esorex [esorex-options] cr2res_util_trace [cr2res_util_trace-options] sof
```

```
DESCRIPTION
  Traces detection
    This utility detects the traces, fits polynomials on their edges
    (Upper and Lower) and in their centers (All), and stores these
    informations in the TRACE_WAVE file.
    Each trace is uniquely identified by its Order/TraceNb values.
    The Order values refer to the keywords indices (e.g. HIERARCH ESO INS
    WLEN CENY4) in the product headers.
    The TraceNb starts with 1, identifies traces within the same order.
    The additional columns :
      Wavelength
      Wavelength_Error
      SlitPolyA
      SlitPolyB
      SlitPolyC
      SlitFraction
    are filled with default values.

    Inputs
      raw.fits FLAT [1 to n]
            or UTIL_CALIB

    Outputs
      <input_name>_tw.fits UTIL_TRACE_TW

    Algorithm
      loop on input raw frames f:
        loop on detectors d:
          Use cr2res_trace(--degree, --min_cluster, --smooth, --opening)
                  to measure the traces
          if --split_traces, call cr2res_trace_split_traces() to split
                 the traces
          Use cr2res_trace_add_extra_columns() to add the additional
                  columns (slit fraction, wl, slit curvature)
        Save the trace wave table

    Library functions used
      cr2res_io_load_image()
      cr2res_trace()
      cr2res_trace_add_extra_columns()
      cr2res_trace_split_traces()
      cr2res_io_save_TRACE_WAVE()


OPTIONS
  The following options are provided by this recipe.

  --degree              : polynomial degree for the fit to the orders. [2]
  --min_cluster         : size in pixels of the smallest allowed cluster.
                          [200000]
  --smooth_x            : Length of the smoothing kernel in x. [111]
  --smooth_y            : Length of the smoothing kernel in y. [401]
  --threshold           : Detection Threshold. [3.0]
  --opening             : Use a morphological opening to rejoin clusters. [TRUE]
  --split_traces        : Split the full slit traces. [0]
  --detector            : Only reduce the specified detector. [0]
```

10.20 cr2res_util_trace_map

```
NAME
  cr2res_util_trace_map -- TRACE_WAVE maps creation

SYNOPSIS
  esorex [esorex-options] cr2res_util_trace_map [cr2res_util_trace_map-options] sof

DESCRIPTION
  Maps creation
    Each input TRACE_WAVE file is converted into maps to visualize the
    traces, wavelengths and the slit curvature

    Inputs
      raw.fits TW [1 to n]

    Outputs
      <input_name>_slit_curve.fits UTIL_TRACE_MAP_SLIT_CURVE
      <input_name>_wave.fits UTIL_TRACE_MAP_WL
      <input_name>_trace.fits UTIL_TRACE_MAP_TRACE

    Algorithm
      loop on input raw frames f:
        loop on detectors d:
          Load the trace_wave extension
          Call cr2res_wave_gen_wave_map() to generate wave_map(d)
          Call cr2res_trace_gen_image() to generate traces_map(d)
          Call cr2res_slit_curv_gen_map() to generate slit_curv_map(d)
      Save wave_map, traces_map and slit_curv_map

    Library Functions used
      cr2res_io_load_TRACE_WAVE()
      cr2res_trace_gen_image()
      cr2res_slit_curv_gen_map()
      cr2res_io_save_SLIT_CURV_MAP()
      cr2res_io_save_WAVE_MAP()
      cr2res_io_save_TRACE_MAP()


OPTIONS
  The following options are provided by this recipe.
  --detector            : Only reduce the specified detector. [0]
  --order               : Only reduce the specified order. [-1]
  --trace_nb            : Only reduce the specified trace number. [-1]
```

10.21 cr2res_util_wave

```
NAME
  cr2res_util_wave -- Wavelength Calibration

SYNOPSIS
  esorex [esorex-options] cr2res_util_wave [cr2res_util_wave-options] sof

DESCRIPTION
  Wavelength Calibration
    This utility performs the wavelength calibration on already extracted
    spectra. It can support different methods (--wl_method parameter):
      XCORR:  Cross Correlation with a emission lines catalog (default)
      LINE1D: Line identification and fitting for each 1D spectra
      LINE2D: Line identification and fitting for all 1D spectra at once
      ETALON: Does not require any static calibration file
      AUTO:   Guess the Method from the input file header

    Inputs
      raw.fits CAL_FLAT_EXTRACT_1D [1 to n]
            or UTIL_EXTRACT_1D
            or UTIL_WAVE_EXTRACT_1D
            or CAL_WAVE_EXTRACT_1D
            or OBS_NODDING_EXTRACTA
            or OBS_NODDING_EXTRACTB
            or OBS_NODDING_EXTRACT_COMB
            or OBS_STARING_EXTRACT
      trace.fits CAL_FLAT_TW [1]
              or CAL_FLAT_TW_MERGED
              or UTIL_TRACE_TW
              or UTIL_WAVE_TW
              or CAL_WAVE_TW
              or UTIL_SLIT_CURV_TW
      lines.fits EMISSION_LINES [0 to 1]

    Outputs
      <input_name>_tw.fits UTIL_WAVE_TW
      <input_name>_wave_map.fits UTIL_WAVE_MAP
      <input_name>_lines_diagnostics.fits UTIL_WAVE_LINES_DIAGNOSTICS
      <input_name>_extracted.fits UTIL_WAVE_EXTRACT_1D

    Algorithm
      loop on raw frames f:
        loop on detectors d:
          Load the trace wave tw(f,d)
          Load the extracted spectra table ext(f,d)
          Call cr2res_wave_apply(tw(f,d),ext(f,d),emission_lines)
              -> lines diagnostics(f,d)
              -> updated_extracted(f,d)
              -> trace_wave_out(f,d)
          Create the wavelength map wave_map(f,d)
        Save lines diagnostics(f)
        Save updated_extracted(f)
        Save wave_map(f)
        Save trace_wave_out(f)

      cr2res_wave_apply()
        loop on the traces t:
          Get the spectrum
          Get the Initial guess
          Switch on the required method:
            CR2RES_LINE2D: cr2res_wave_2d()
            CR2RES_LINE1D: cr2res_wave_line_fitting()
            CR2RES_ETALON: cr2res_wave_etalon()
            CR2RES_XCORR:  cr2res_wave_xcorr()

    Library Functions used
      cr2res_io_find_TRACE_WAVE()
      cr2res_io_load_TRACE_WAVE()
      cr2res_io_load_EXTRACT_1D()
      cr2res_wave_apply()
      cr2res_extract_EXTRACT1D_get_spectrum()
      cr2res_wave_estimate_compute()
      cr2res_wave_2d()
      cr2res_wave_line_fitting()
      cr2res_wave_etalon()
      cr2res_wave_xcorr()
      cr2res_wave_gen_wave_map()
      cr2res_io_save_TRACE_WAVE()
      cr2res_io_save_WAVE_MAP()
      cr2res_io_save_LINES_DIAGNOSTICS()
      cr2res_io_save_EXTRACT_1D()


OPTIONS
  The following options are provided by this recipe.

  --detector            : Only reduce the specified detector. [0]
  --order               : Only reduce the specified order. [-1]
  --trace_nb            : Only reduce the specified trace number. [-1]
  --wl_method           : Wavelength Method (XCORR / LINE1D / LINE2D / ETALON).
                          [UNSPECIFIED]
  --wl_shift            : Wavelength shift (nm) to apply to the guess. [0.0]
  --wl_est              : Estimated wavelength [start, end] (in nm). [-1.0,
                        -1.0]
  --wl_err            : Estimated wavelength error (in nm). [-1.0]
  --wl_degree         : Wavelength Polynomial degree, main dispersion. [5]
  --wl_xdegree        : Wavelength Polynomial degree, cross-dispersion. [5]
  --log               : Flag for taking the Log() value of the lines. [FALSE]
  --fallback          : Flag for using the input WL when no computation.
                        [FALSE]
  --keep              : Flag for re-using higher degrees of first guess in
                        Cross-Corr. [FALSE]
  --clean_spectrum    : Clean spectrum to use only around catalogue lines.
                        [TRUE]
  --display           : Flag for display. [FALSE]
  --display_range     : Wavelength range to display [start, end] (in nm).
                        [-1.0, -1.0]
```

11 Data

11.1 Raw Data

DPR.TYPE	DPR.CATG	Processed by
DARK	CALIB	cr2res_cal_dark
FLAT	CALIB	cr2res_cal_flat
LAMP,METROLOGY	CALIB	None
WAVE,ABSORPTION_SGC	CALIB	None, atm.
WAVE,ABSORPTION_N2O	CALIB	None, atm.
WAVE,ABSORPTION-CELL	CALIB	None
WAVE,UNE	CALIB	cr2res_cal_wave
WAVE,FPET	CALIB	cr2res_cal_wave
WAVE,LAMP	CALIB	None.
FLAT,LAMP,DETCHECK	CALIB	cr2res_cal_detlin
DARK,DETCHECK	CALIB	cr2res_cal_detlin

The following table lists the files with DPR.CATG=SCIENCE.

DPR.TYPE	DPR.TECH	Processed by
OBJECT	SPECTRUM,DIRECT,OTHER	cr2res_obs_staring
OBJECT	SPECTRUM,NODDING,JITTER	cr2res_obs_nodding
OBJECT	SPECTRUM,NODDING,OTHER	cr2res_obs_nodding
OBJECT	SPECTRUM,NODDING,OTHER,ASTROMETRY	cr2res_obs_nodding
OBJECT	SPECTRUM,NODDING,OTHER,POLARIMETRY	cr2res_obs_pol
OBJECT	SPECTRUM,GENERIC	cr2res_obs_2d
SKY	SPECTRUM,GENERIC	cr2res_obs_2d
STD	SPECTRUM,NODDING,OTHER,POLARIMETRY	cr2res_obs_pol
STD	SPECTRUM,NODDING,OTHER	cr2res_obs_nodding

In addition to the recipes listed in the tables, any raw file can also be processed by cr2res_util_calib to apply calibrations or combine frames.

11.2 Data products

The following table lists the DRS data products. The PRO.TYPE defines the format and content of a file, and PRO.CATG name convention tells which recipe the file was made from.

PRO.TYPE	PRO.CATG	Remarks
BPM	CAL_DARK_BPM	
BPM	CAL_DETLIN_BPM	
BPM	CAL_FLAT_BPM	
BPM	UTIL_NORM_BPM	
CALIBRATED	UTIL_CALIB	
COMBINED	OBS_NODDING_COMBINEDA	
COMBINED	OBS_NODDING_COMBINEDB	
DETLIN_COEFFS	CAL_DETLIN_COEFFS	
EXTRACT_1D	CAL_FLAT_EXTRACT_1D	
EXTRACT_1D	CAL_WAVE_EXTRACT_1D	
EXTRACT_1D	OBS_NODDING_EXTRACTA	
EXTRACT_1D	OBS_NODDING_EXTRACTB	
EXTRACT_1D	OBS_NODDING_EXTRACT_COMB	
EXTRACT_1D	UTIL_EXTRACT_1D	
EXTRACT_1D	UTIL_WAVE_EXTRACT_1D	
EXTRACT_2D	OBS_2D_EXTRACT	
LINES_DIAGNOSTICS	CAL_WAVE_LINES_DIAGNOSTICS	
LINES_DIAGNOSTICS	UTIL_WAVE_LINES_DIAGNOSTICS	
MASTER_DARK	CAL_DARK_MASTER	
MASTER_FLAT	CAL_FLAT_MASTER	
MASTER_FLAT	UTIL_MASTER_FLAT	
PRO.TYPE	PRO.CATG	
SLIT_CURV_MAP	UTIL_SLIT_CURV_MAP	
SLIT_FUNC	CAL_FLAT_SLIT_FUNC	
SLIT_FUNC	OBS_NODDING_SLITFUNCA	
SLIT_FUNC	OBS_NODDING_SLITFUNCB	
SLIT_FUNC	UTIL_SLIT_FUNC	
SLIT_MODEL	CAL_FLAT_SLIT_MODEL	
SLIT_MODEL	OBS_NODDING_SLITMODELA	
SLIT_MODEL	OBS_NODDING_SLITMODELB	
SLIT_MODEL	UTIL_SLIT_MODEL	
TW	CAL_FLAT_TW	
TW	CAL_FLAT_TW_MERGED	
TW	CAL_WAVE_TW	
TW	OBS_NODDING_TWA	
TW	OBS_NODDING_TWB	
TW	UTIL_SLIT_CURV_TW	
TW	UTIL_TRACE_TW	
TW	UTIL_WAVE_TW	
WAVE_MAP	CAL_WAVE_MAP	
WAVE_MAP	UTIL_WAVE_MAP	
CATALOG	EMISSION_LINES	from `cr2res_util_genlines`
PHOTO_FLUX	PHOTO_FLUX	from `cr2res_util_genstd`

Data products table continued:

PRO.TYPE	PRO.CATG	Remarks
POL_SPEC	OBS_POL_SPECA	
POL_SPEC	OBS_POL_SPECB	
EXTRACT_1D	OBS_POL_EXTRACTA	optional product
EXTRACT_1D	OBS_POL_EXTRACTB	optional product
TW	OBS_POL_TWA	optional product
TW	OBS_POL_TWB	optional product
CALIBRATED	OBS_POL_CALIBA	optional product
CALIBRATED	OBS_POL_CALIBB	optional product

The data products of `cr2res_obs_nodding` are saved in files which are named with suffixes as follows.

- `_OBS_NODDING_COMBINED[AB]` are the combined frames from A and B, one minus the other, i.e. one positive and one negative image, before extraction.
- `_OBS_NODDING_TW[AB].fits` are the TW tables for extraction of A or B, respectively. These are made from the input TW, depending on the NODTHROW.
- `_OBS_NODDING_EXTRACT[AB].fits` are the main data products, the extracted spectra from nodding positions A and B.
- `_OBS_NODDING_EXTRACT_COMB.fits` is the last two combined, i.e. one re-sampled on the WL-scale of the other, and added together.
- `_OBS_NODDING_SLITFUNC[AB].fits` is the (oversampled) slit-illumination function (for diagnostics only).
- `_OBS_NODDING_SLITMODEL[AB].fits` is the 2D frame reconstructed from the slit-function and the spectrum (for diagnostics only).

Here, `[AB]` is shorthand for "either one of `A` or `B`", i.e. the nodding positions.

The optional intermediate data products for polarimetry are named in a similar fashion.

A Installation

ESO pipelines can be installed via several methods, depending on your OS, most of which facilitate easy installation, upgrade and removal. Please see the *ESO Data Reduction Pipelines and Workflow Systems* page (*https://www.eso.org/pipelines*).

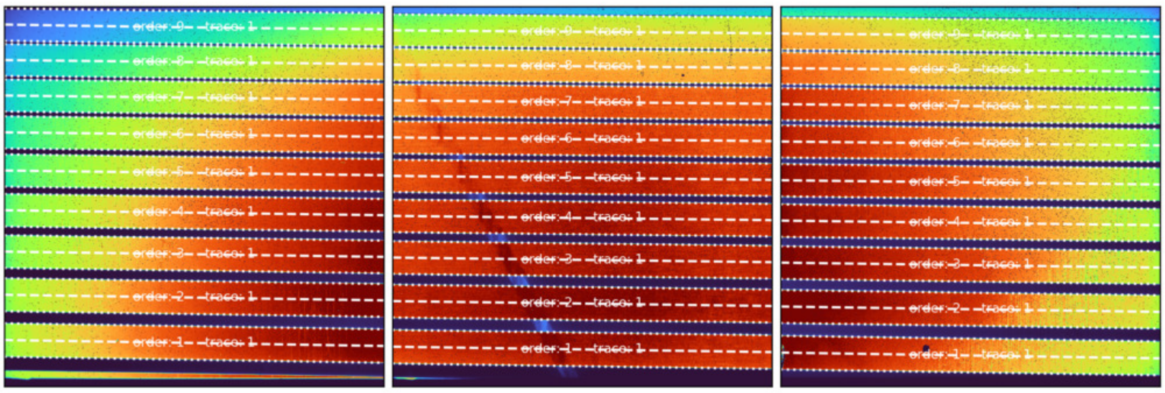
B Additional figures

B.1 Traces and slit tilt all standars settings

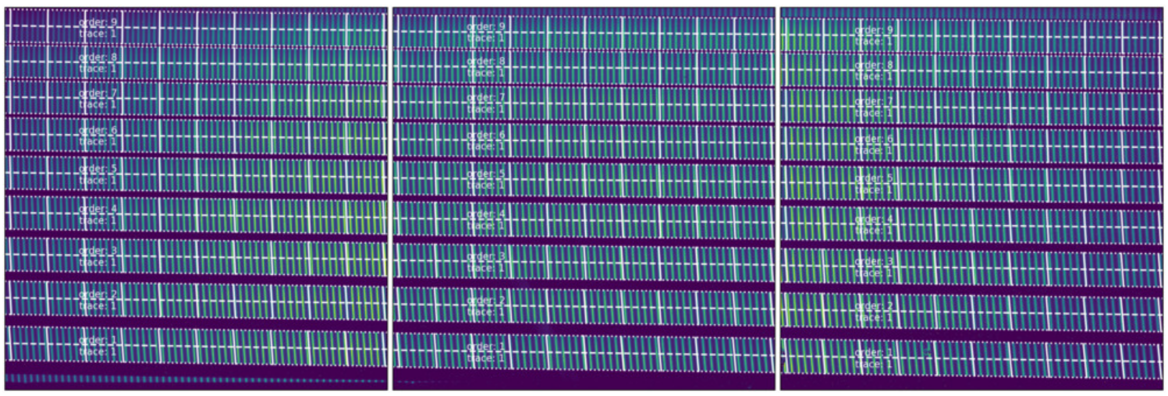
Figures for each of the standard settings, analogous to Fig. 4.2. The background image is a halogen-lamp exposure, white lines are the polynomials of the order trace in the TW table, i.e. the mid-line (dashed) and upper/lower edges (dotted).

In the bands YJHK, a second plot shows an FPET frame as background image, again with the traces overplotted, but now also the tilted slit projection, as regularly spaced, solid white lines.

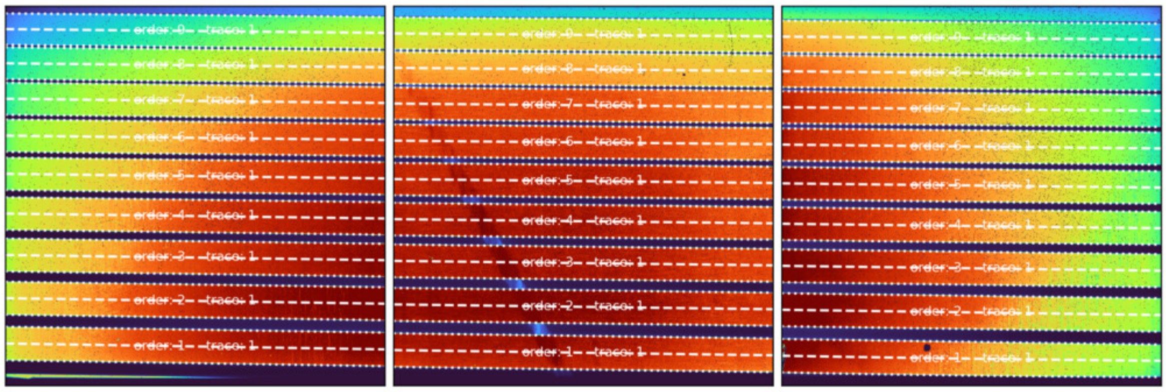
*

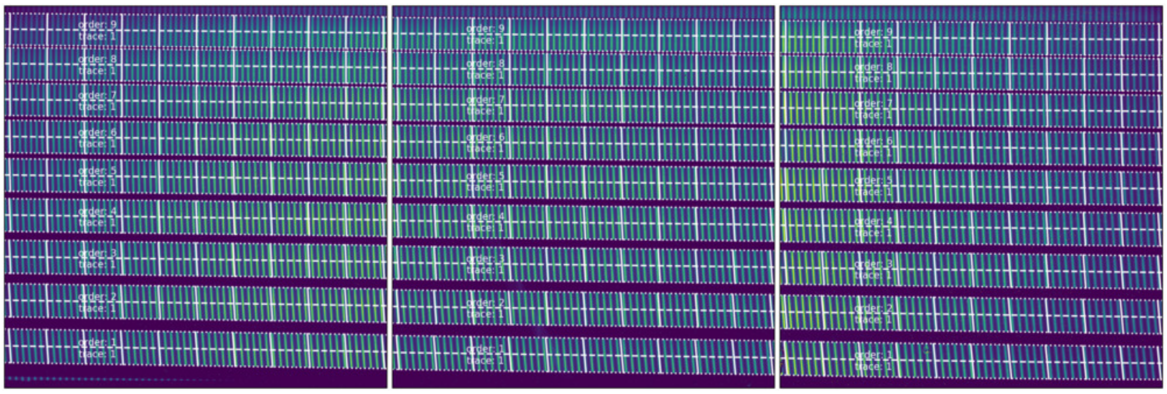


Y1028

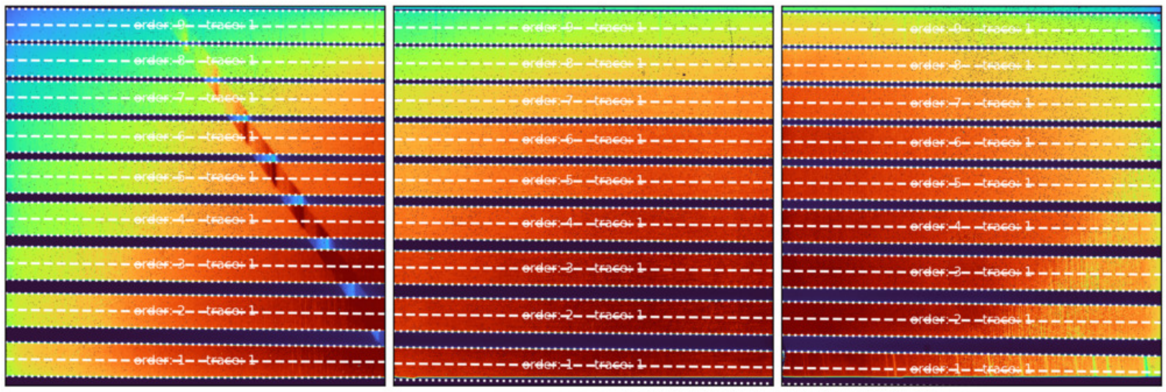


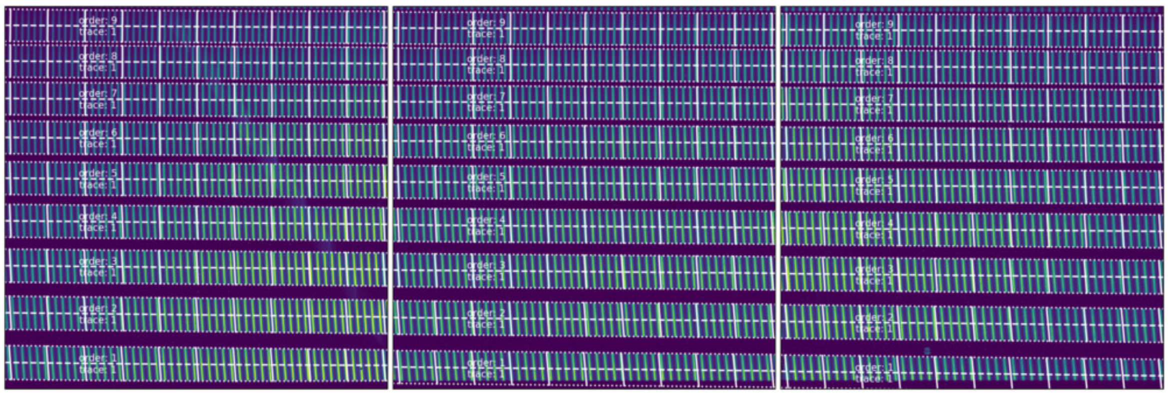
Y1029


order: 9 trace: 1
order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1
order: 1 trace: 1

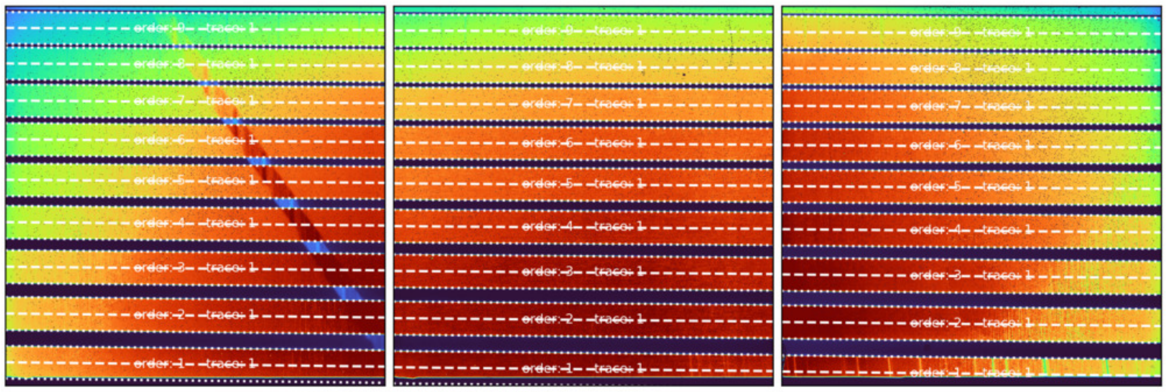


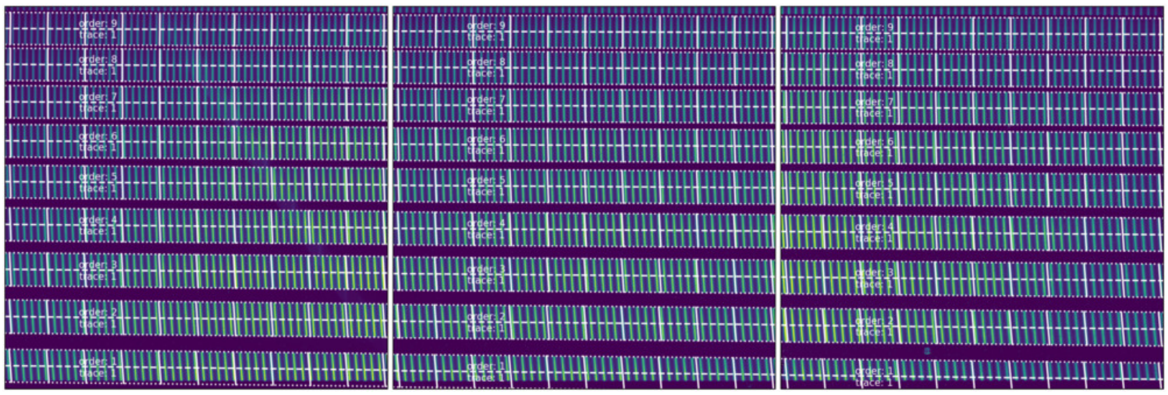
J1226


order: 9 trace: 1
order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1
order: 1 trace: 1

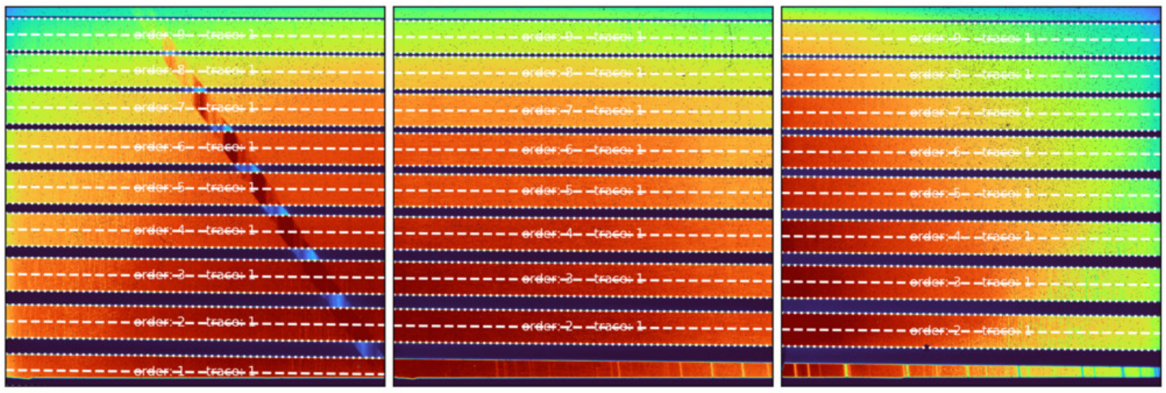


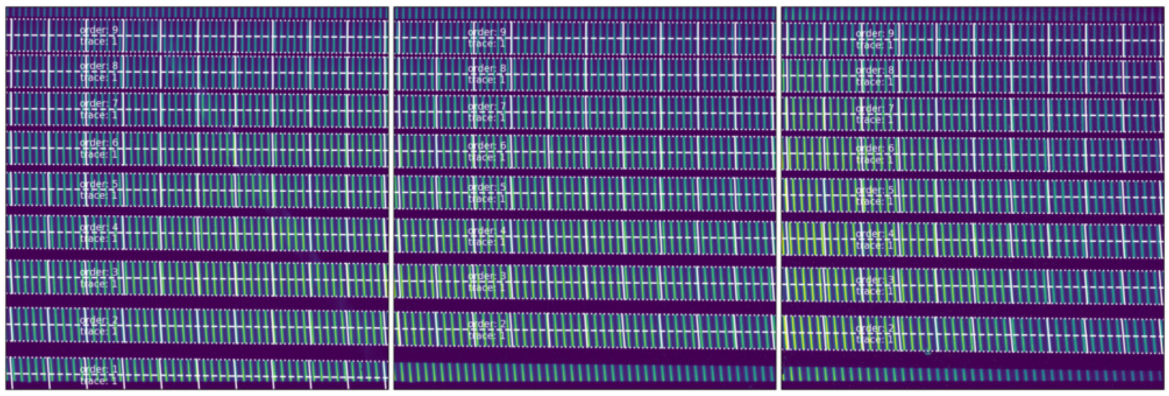
J1228



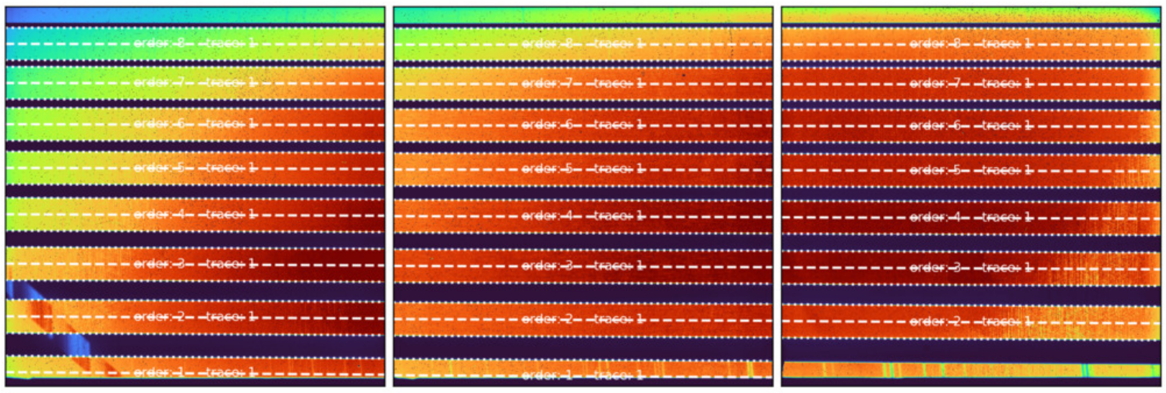


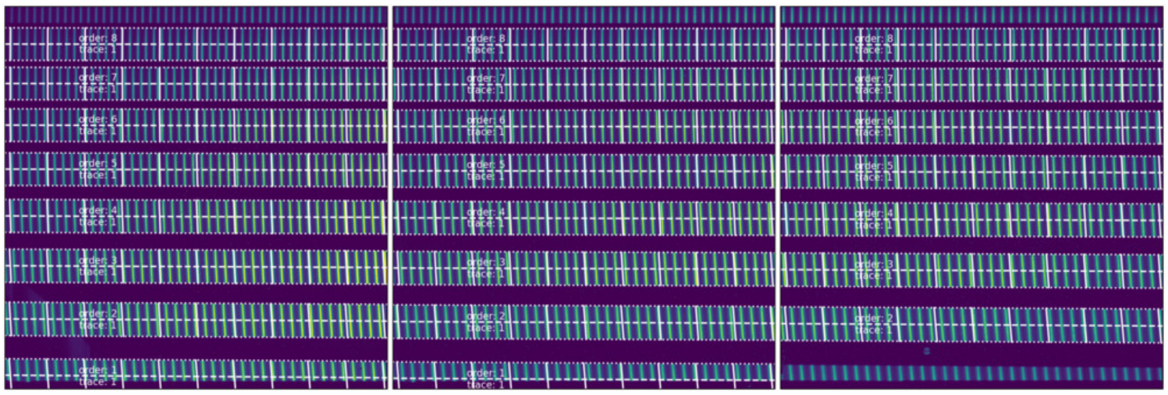
J1232


order: 9 trace: 1
order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1
order: 1 trace: 1

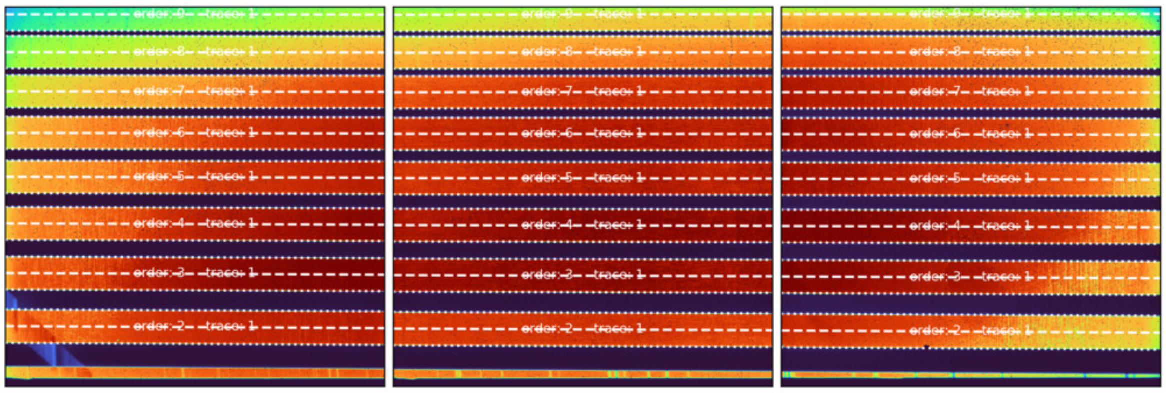

order: 9 trace: 1
order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1
order: 1 trace: 1

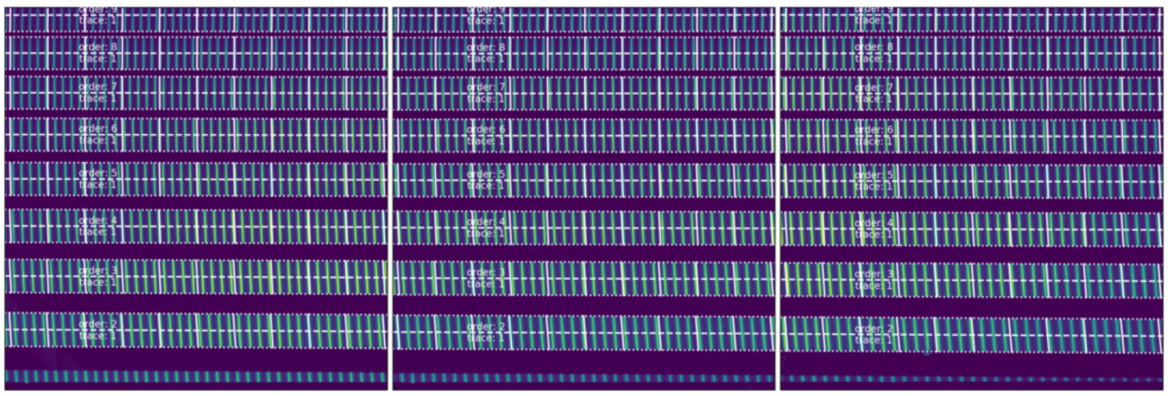
H1559


order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1
order: 1 trace: 1

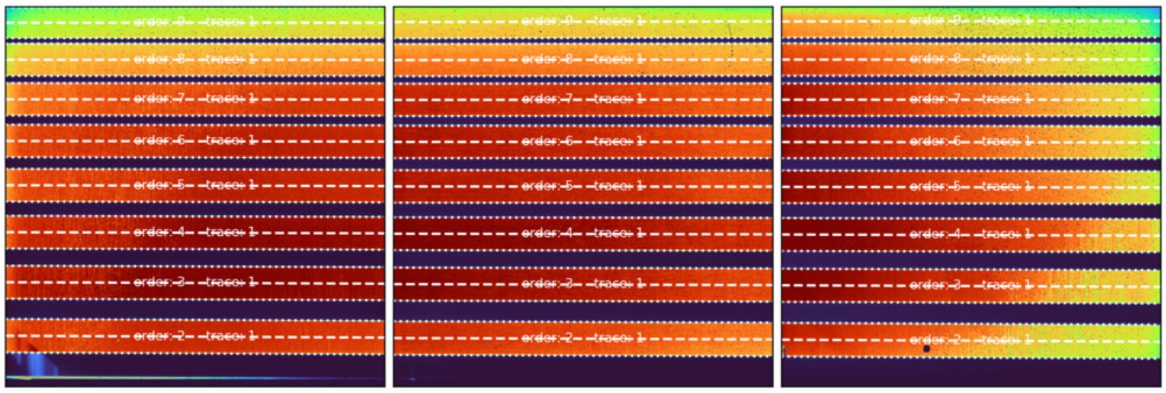


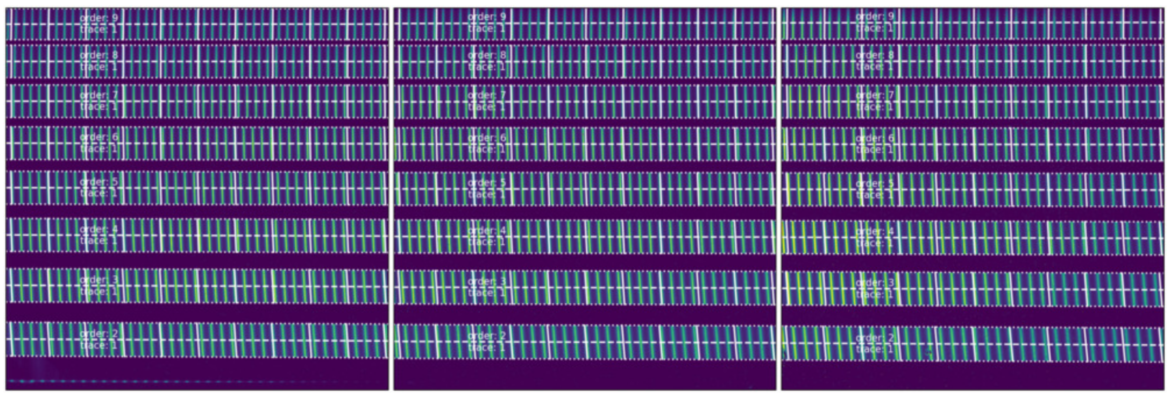
H1567



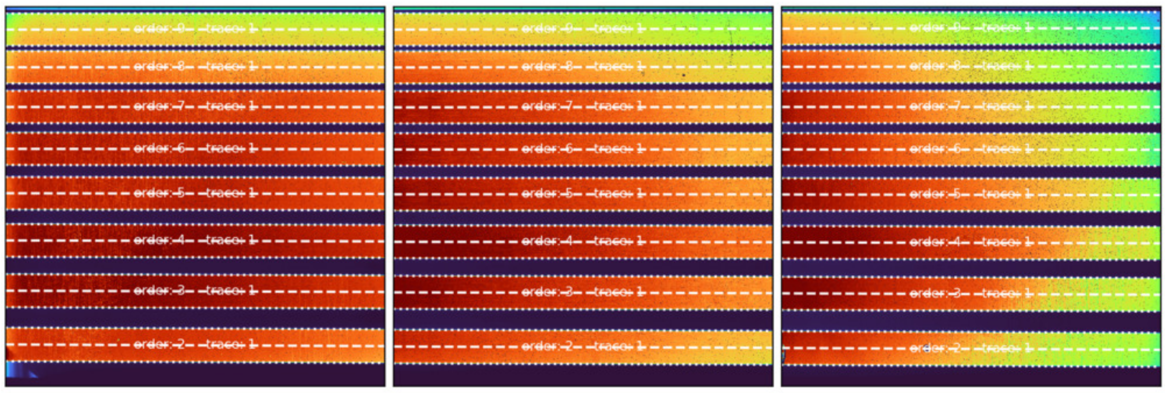


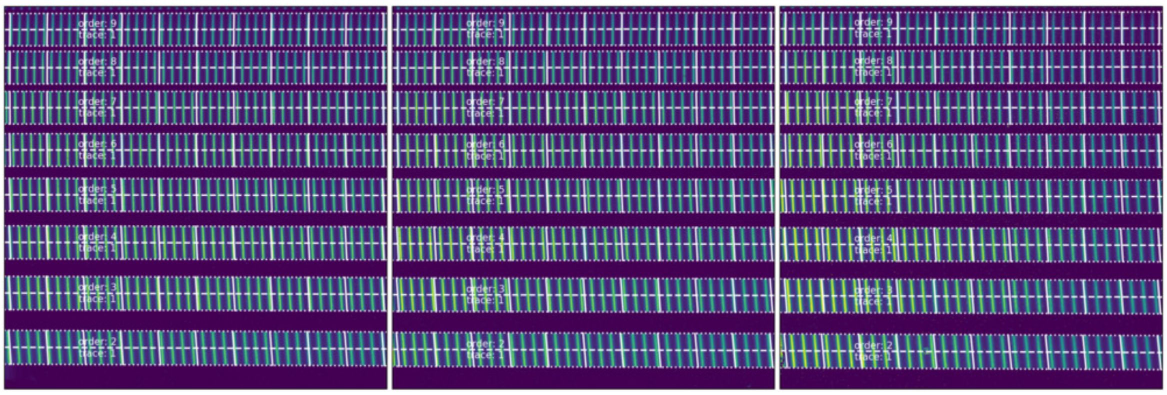
H1575


order: 9 trace: 1
order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1

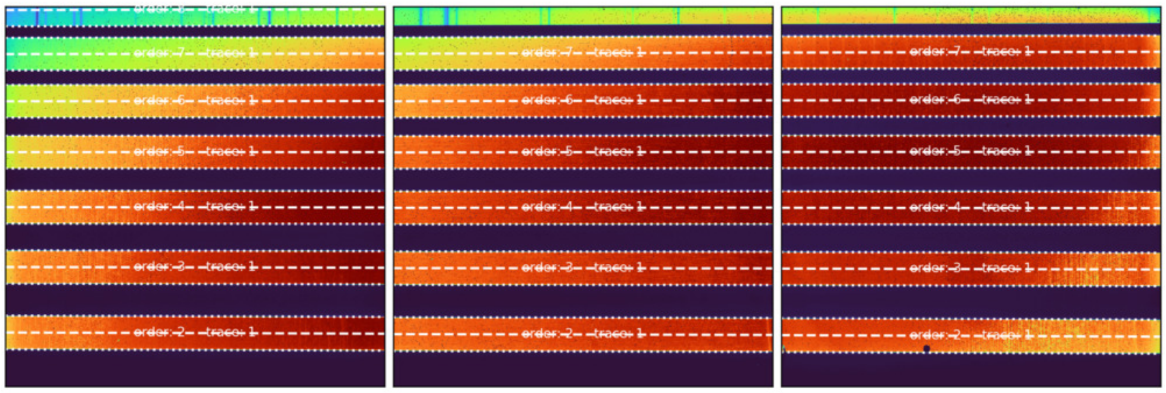

order: 9 trace: 1
order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1

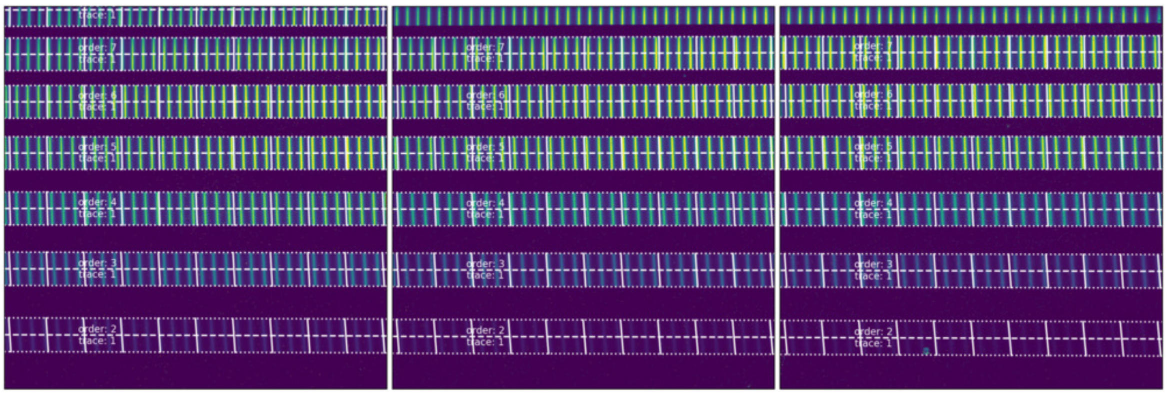
H1582


order: 9 trace: 1
order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1

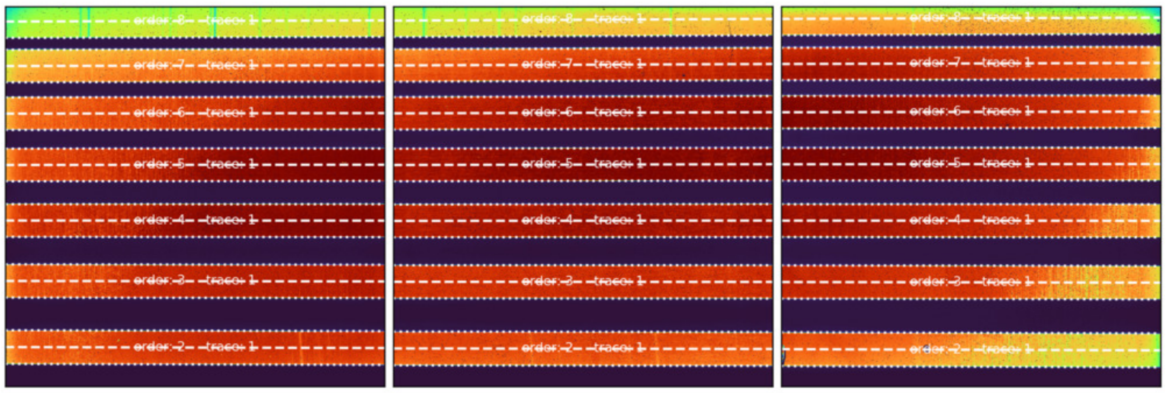

order: 9
trace: 1
order: 8
trace: 1
order: 7
trace: 1
order: 6
trace: 1
order: 5
trace: 1
order: 4
trace: 1
order: 3
trace: 1
order: 2
trace: 1

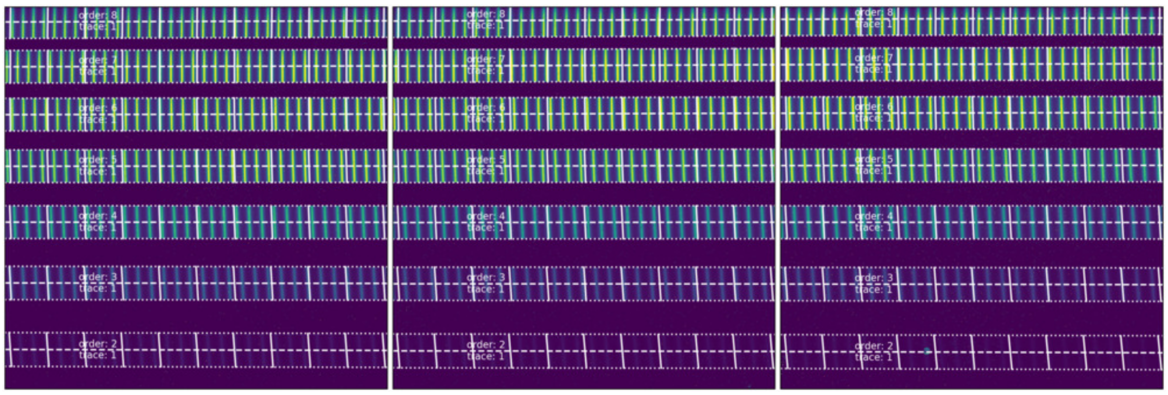
K2148


order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1

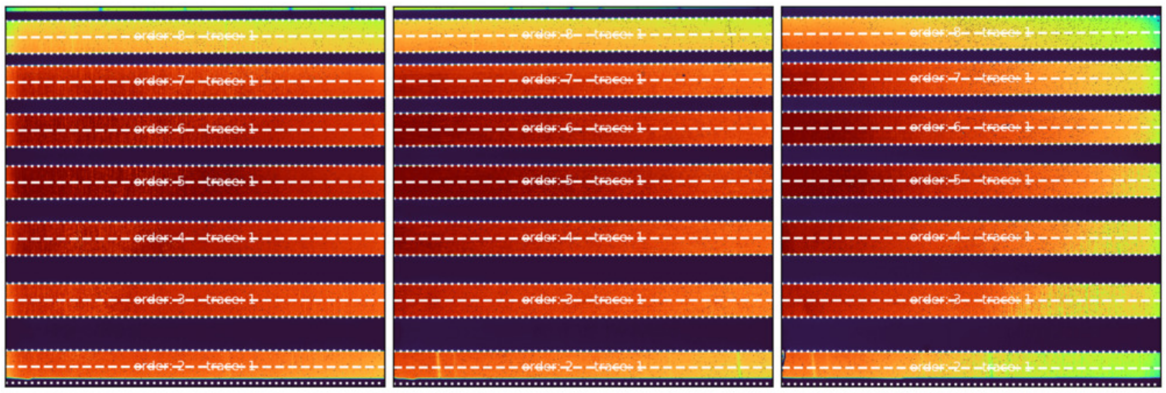

order: 7
trace: 1
order: 6
trace: 1
order: 5
trace: 1
order: 4
trace: 1
order: 3
trace: 1
order: 2
trace: 1

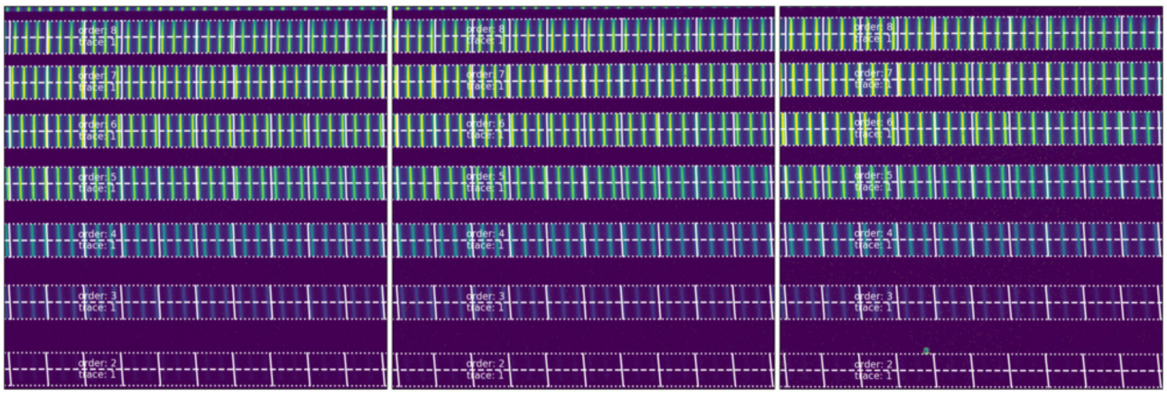
K2166


order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1

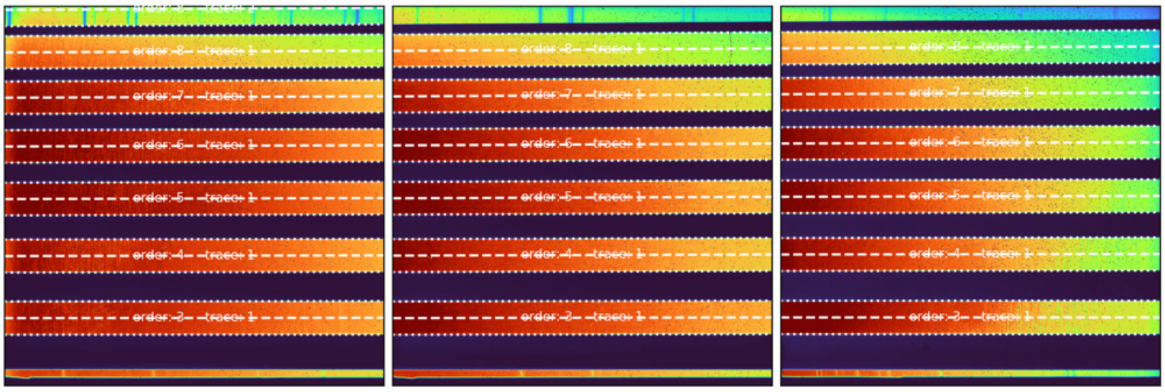

order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1

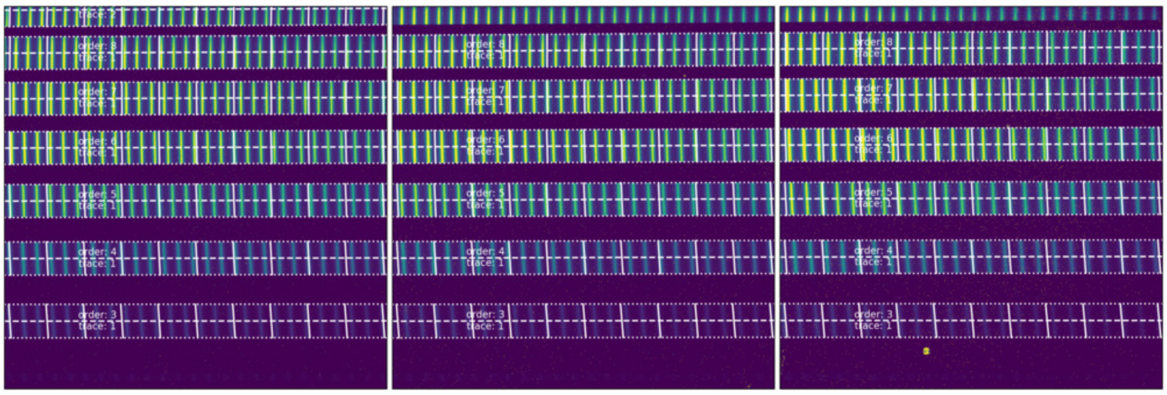
K2192


order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1

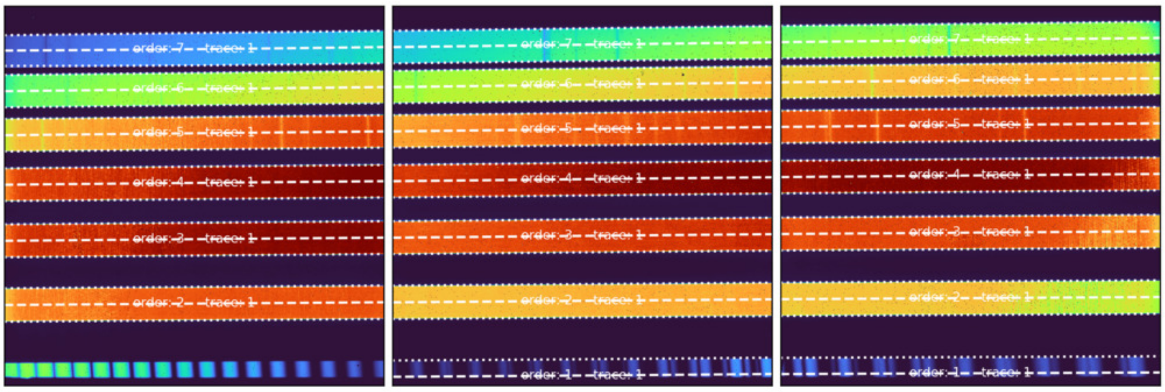

order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1
order: 2 trace: 1

K2217

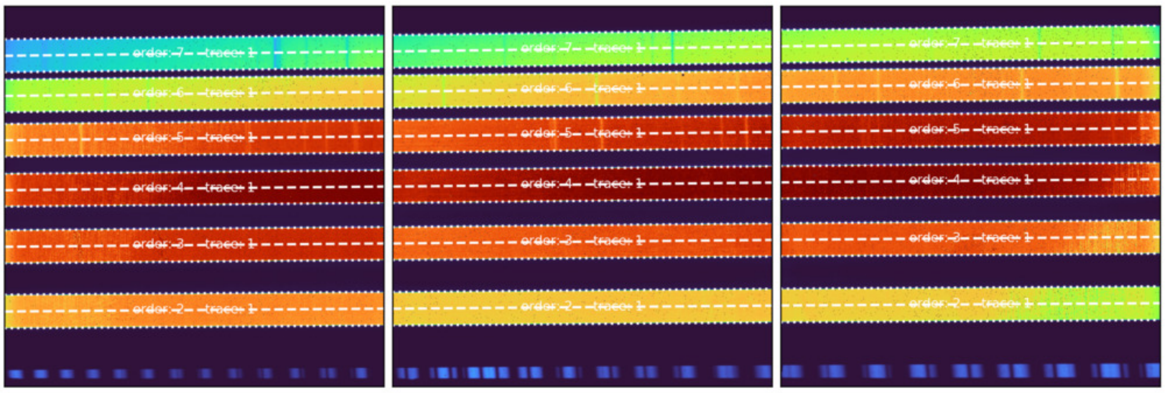

order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1


order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1

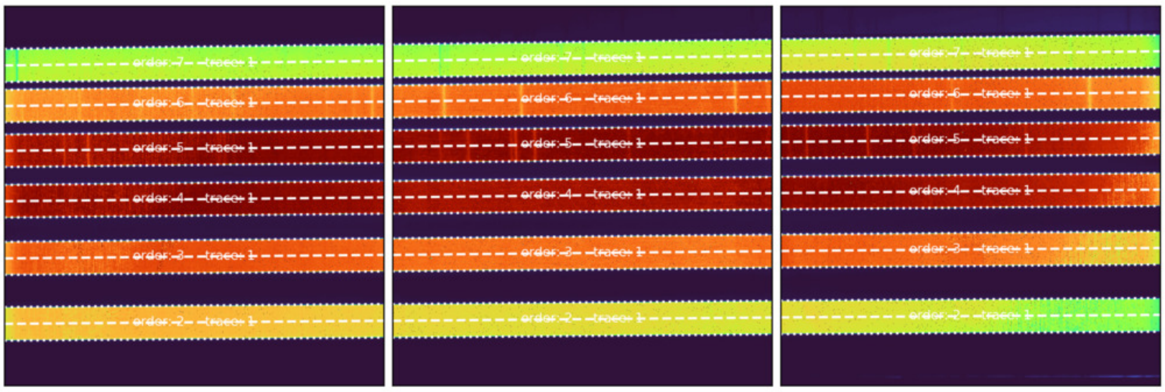
L3244



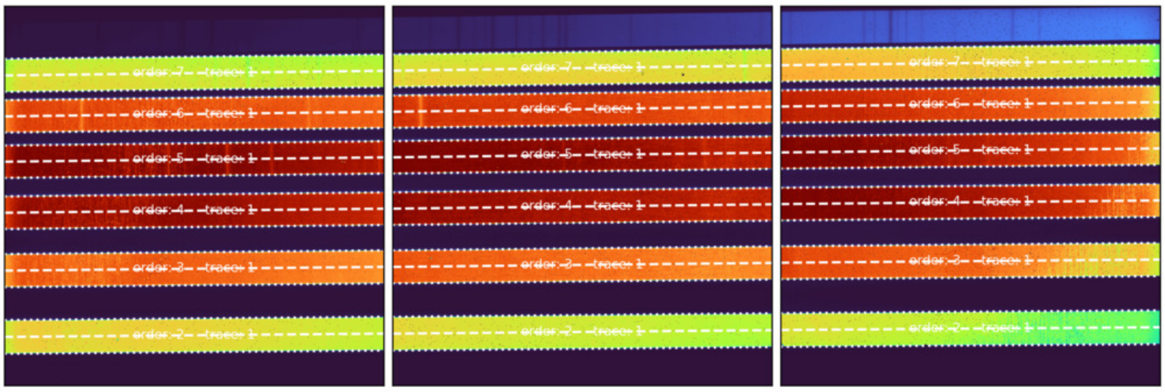
L3262



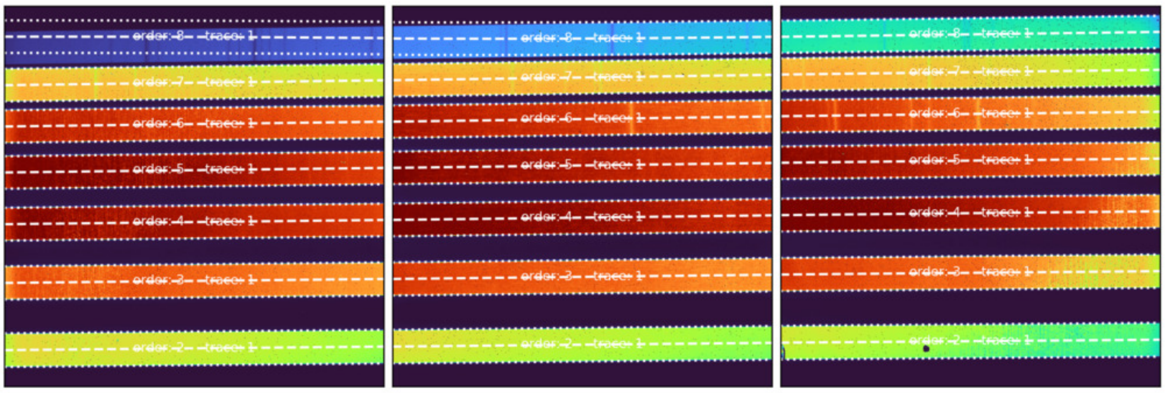
L3302



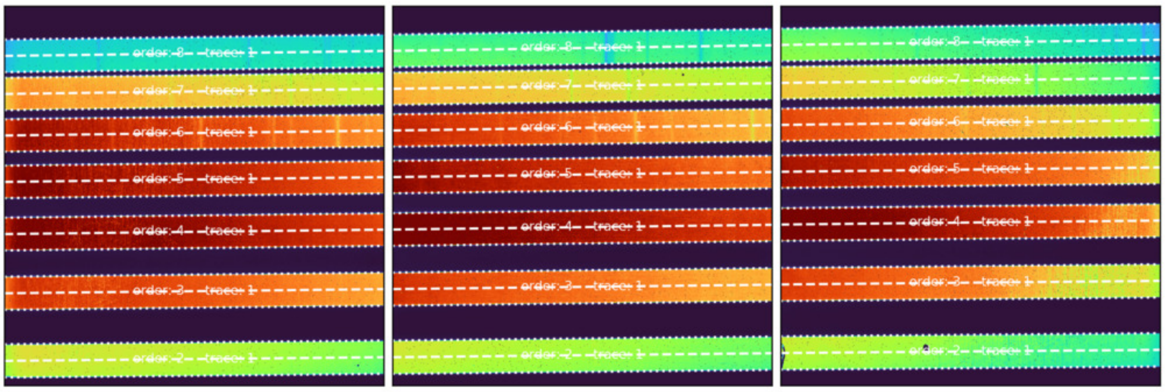
L3340



L3377

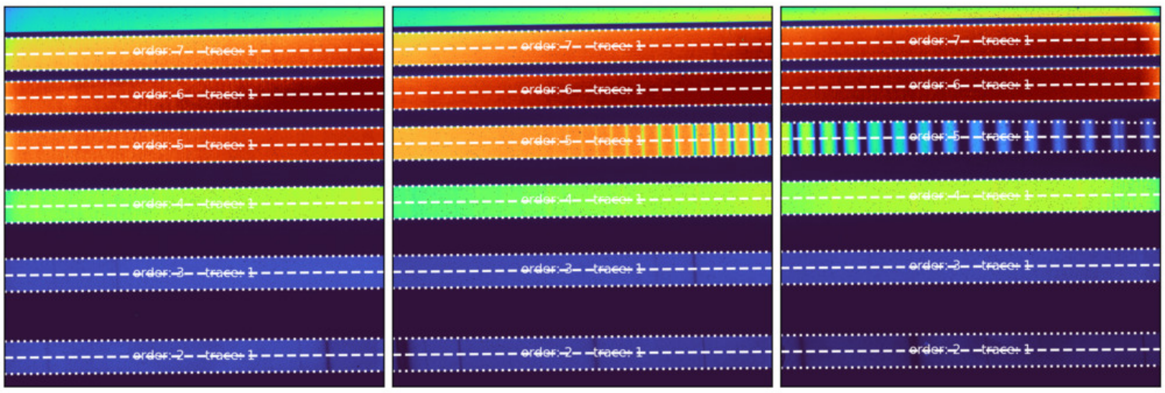


L3412

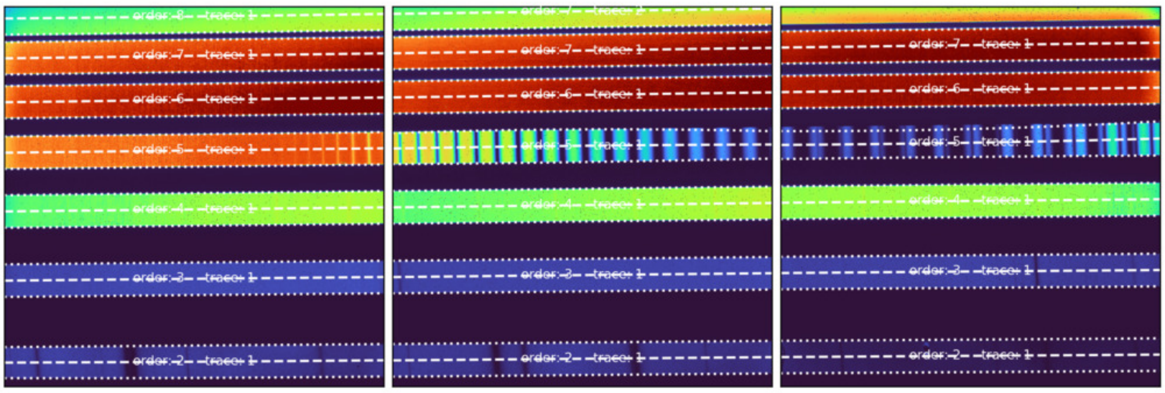


L3426

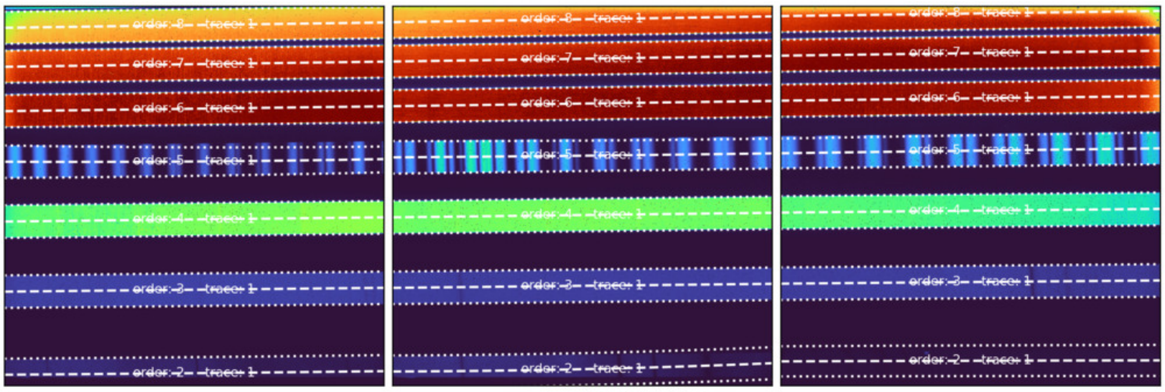
M4187



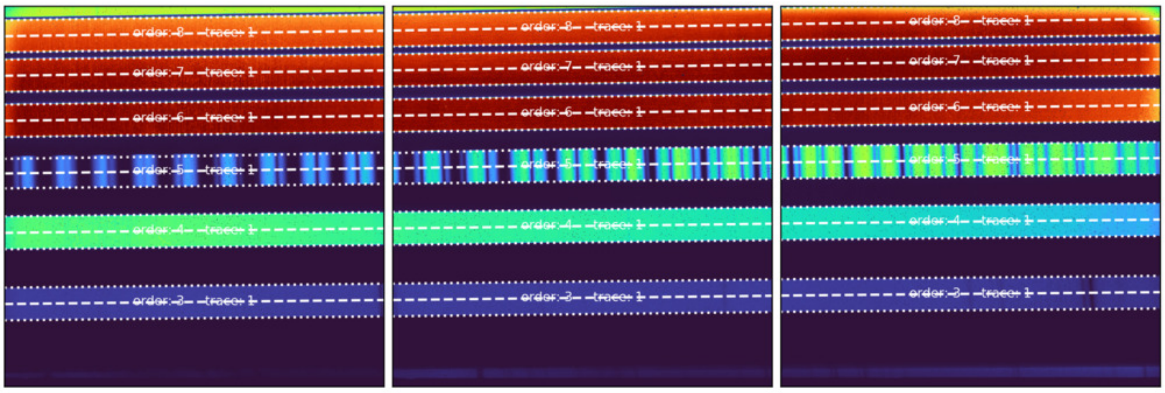
M4211



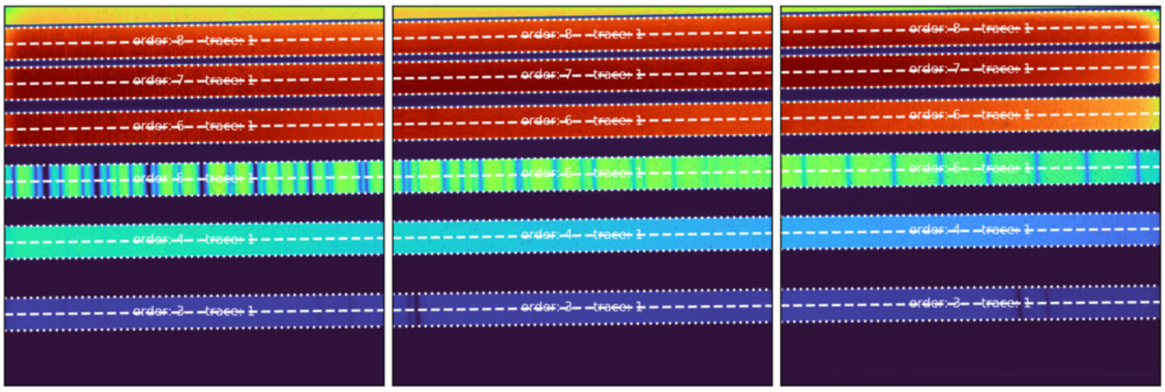
M4266



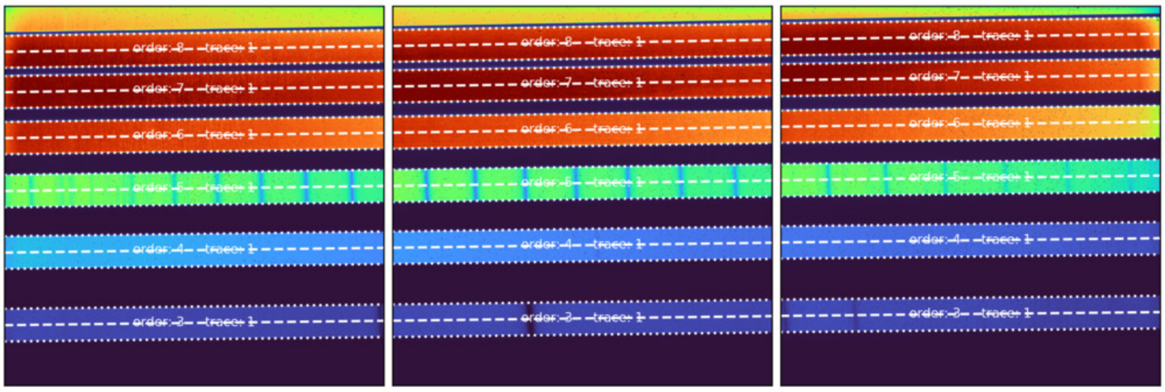
M4318



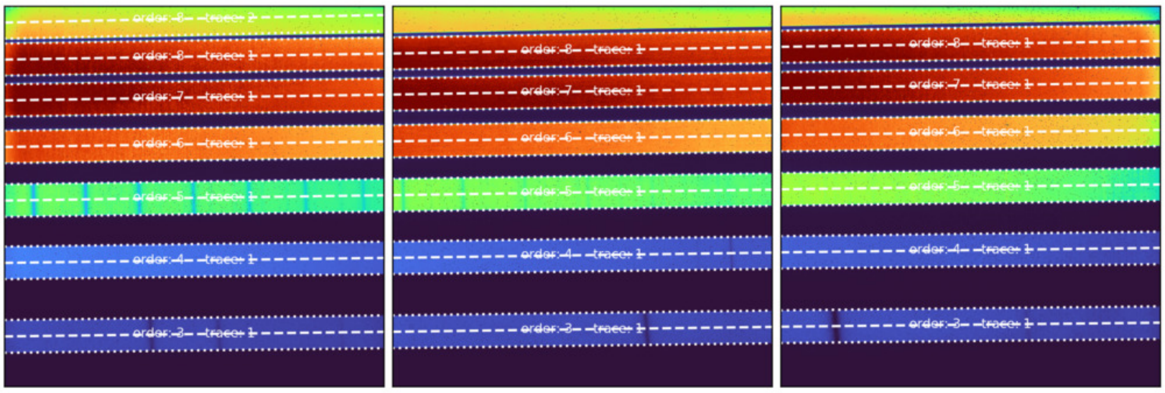
M4368



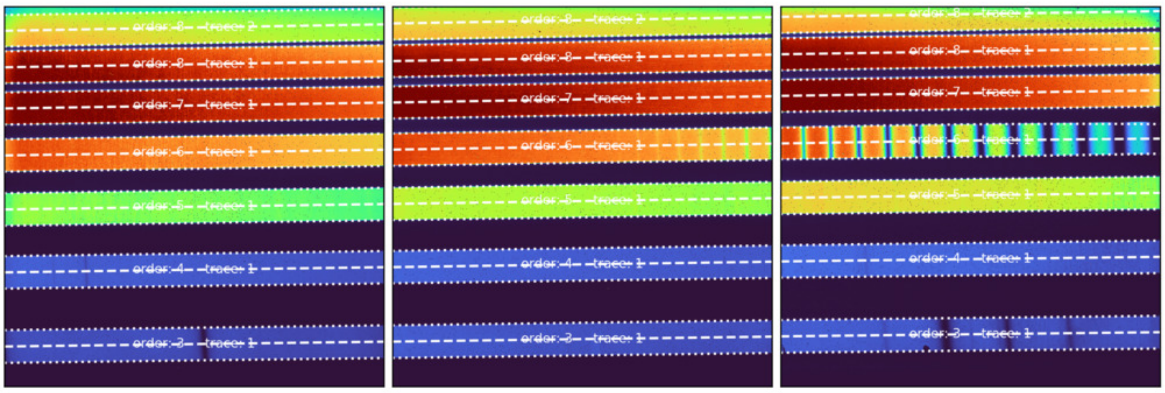
M4416



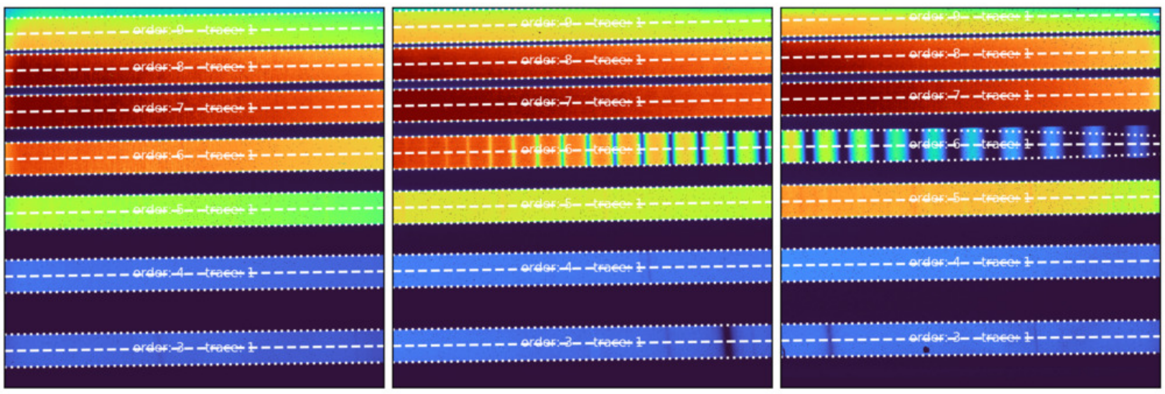
M4461



M4504



M4519


order: 9 trace: 1
order: 8 trace: 1
order: 7 trace: 1
order: 6 trace: 1
order: 5 trace: 1
order: 4 trace: 1
order: 3 trace: 1

C Processing reduced CRIRES data with MOLECFIT

Introduction

Telluric lines - absorption from species in the Earth atmosphere - are strongly contaminating astronomical spectroscopic observation in the near-infrared, being superimposed to the spectrum of the science target. The removal of telluric lines from observations is therefore an important step in the reduction and analysis of near-infrared spectra .

The removal of telluric lines is usually done either by observing a telluric standard star close in time from the science target, or through the modelling of the telluric spectrum. The latter method has the advantage of using no observing time. One tool of interest for ESO instruments which can model telluric spectrum and correct it in science spectra is `MOLECFIT` [10] [10].

The processing of CRIRES extracted spectra with `MOLECFIT` is, at this point, neither a part of the DRS nor supported by ESO. However, we provide here basic instructions and a custom script to convert CRIRES reduced spectra into data which are ready to be processed with `MOLECFIT`. This appendix is not a tutorial, we assume that the user is familiar with `MOLECFIT`.

Instructions

We assume that the user has an extracted spectrum from the DRS, hereafter called input spectrum. It can be a spectrum:

- acquired in staring or nodding mode
- if from nodding: either from A or B position, or combined

The python script `cr2res_drs2molecfit` will change the format of the input spectrum so that it can be processed with `MOLECFIT`. The primary extension of the output spectrum will be identical to the input spectrum. However, every subsequent FITS extention will contain a spectral order projected on a single detector in table format with threw columns: WAVE, SPEC, and ERR containing respectively the wavelength in microns, the spectrum, and the error spectrum.

The script `cr2res_drs2molecfit` also creates a corresponding

- a `WAVE_INCLUDE.fits` file, indicating the wavelength region to be included in the fit. By default, all the spectrum is included, the user can modify the file if they wish.
- a `ATM_PARAMETERS_EXT.fits` file

Make sure to edit the configuration file for the `molecfit_model` recipe with:

- `COLUMN_LAMBDA=WAVE`

[10] `MOLECFIT` can be downloaded at `https://www.eso.org/sci/software/pipelines/`

- `COLUMN_FLUX=SPEC`
- `COLUMN_DFLUX=ERR`

so that `MOLECFIT` can map the columns from the SCIENCE.fits file to wavelength, flux, and error arrays.

D UNE line selection

Storing and using line selections

Static calibration files containing UNE line lists are distributed with the pipeline, and are named like `lines_u_redman_H1575.fits`. These are the input to `cr2res_cal_wave` or `cr2res_util_wave` and contain a subset of lines from the full catalogs by [9] (for Y and J bands) and [7] (for H and K). Each subset has been manually selected to only contain usable lines for the current setting.

The raw catalogs are part of *cr2rep*, as txt-files in the subdirectory *catalogs/*, containing the wavelength and strength of the lines. In the subdirectory *selections/* there are further txt-files, one for each setting. They contain one *usable wavelength region* per line, i.e. start and end wavelength between which there are only usable lines in the full catalog. This can be a single line, a group of lines, or a whole detector-order; depending on how well the catalog locally matches the obtained spectra.

The recipe `cr2res_util_genlines` then takes the catalogs and selection files to generate the FITS-files above.

The following describes how to go about selecting lines. This might become necessary when the UNE lamp is exchanged[11] or has aged, or when for some other reason the cross-correlation between spectrum and catalog no longer gives a reliable zero-point to the wavelength solution.

Selecting lines

Earlier attempts had been made to select lines by objective criteria, or by analyzing spectra from other lamps or instruments. In practice however, it turned out that these were both insufficient in wavelength coverage and “cleanliness” of the lines (in terms of blends, ghosts etc.).

What worked well when we started to calibrate CRIRES**+**, was to visually compare the spectra to the catalog in a plot like Fig. D.1. It becomes apparent quickly from the relative line strengths, which lines in the spectrum are the catalog lines.

There are several sources of ambiguity and things to keep in mind when selecting lines:

- There are lines in the spectrum that are not in the catalogs. This obviously is true for the strong Ne and Ar lines, but also for plenty of weak lines.
- Not all lines in the catalogs are present in the spectra, even among the stronger catalog lines.
- The default method applied by the wavecal recipes is cross-correlation with a synthetic spectrum that gets generated from the selected lines. It therefore makes sense to select lines with comparable strengths within the same detector order. At least avoid the case of a single line dominating the correlation.
- This is not possible for all wavelength regions. Especially in the K-band, lines become so sparse that you have to take what you get.

[11] Yes, they can be surprisingly different.

- Blended lines should be avoided, if possible. When adjacent lines are both in the catalog, with matching relative strengths, they can usually be included.

Fig. D.1 was made with the script `cr2res_show_spec_catal.py` that is included in the pipeline source code, subdirectory *tools/*. It provides a rudimentary user interface that allows to print the wavelength under the cursor, thereby writing the selection file; see source code for details. Of course, many other tools can be used to do the simple task of plotting and reading-off wavelengths.

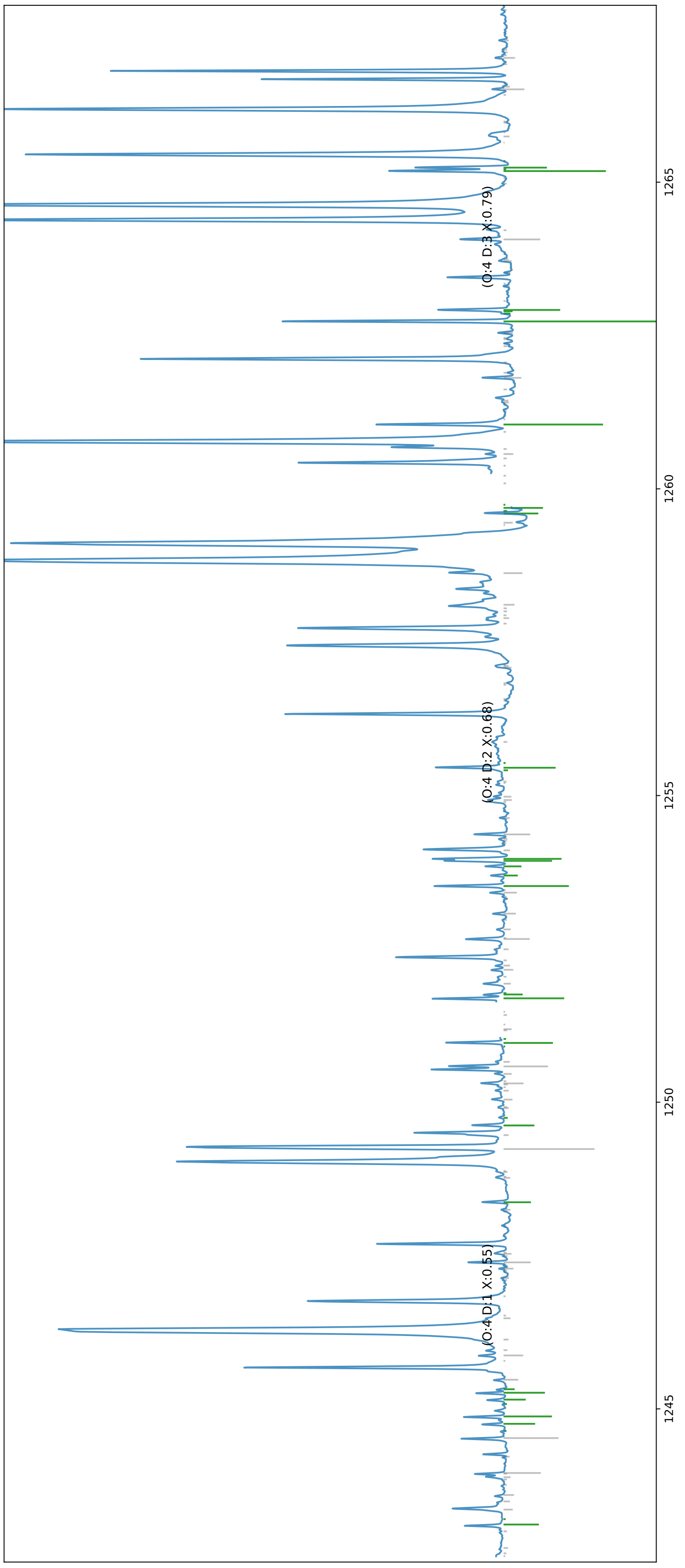


Figure D.1: An example of a UNe spectrum in setting J1228. The catalog lines are plotted upside down, in green the selected lines, in gray the not-selected ones.

[1] F. Cersullo, A. Coffinet, B. Chazelas, C. Lovis, and F. Pepe. New wavelength calibration for echelle spectrographs using Fabry-Pérot etalons. *Astronomy and Astrophysics*, 624:A122, April 2019. 10, 43

[2] J. F. Donati, M. Semel, B. D. Carter, D. E. Rees, and A. Collier Cameron. Spectropolarimetric observations of active stars. *MNRAS*, 291(4):658–682, November 1997. 10, 30

[3] J. Smoker et. al. *oCRIRES Data Reduction Cookbook*. ESO, https://www.eso.org/sci/facilities/paranal/instruments/crires/doc/VLT-MAN-ESO-14200-4032_v91.pdf, 91.0 edition, 2012. VLT-MAN-ESO-14200-4032. 9, 10

[4] Nausicaa Delmotte et al. *ESO Science Data Products Standard*. ESO, https://www.eso.org/sci/observing/phase3/p3sdpstd.pdf, 8 edition, 2022. ESO-044286. 17

[5] K. Horne. An optimal extraction algorithm for CCD spectroscopy. *PASP*, 98:609–617, June 1986. 41

[6] Nikolai Piskunov, Ansgar Wehrhahn, and Thomas Marquart. Optimal extraction of echelle spectra: Getting the most out of observations. *Astronomy and Astrophysics*, 646:A32, February 2021. 10, 12, 37

[7] Stephen L. Redman, James E. Lawler, Gillian Nave, Lawrence W. Ramsey, and Suvrath Mahadevan. The Infrared Spectrum of Uranium Hollow Cathode Lamps from 850 nm to 4000 nm: Wavenumbers and Line Identifications from Fourier Transform Spectra. *Astrophysical Journal, Supplement*, 195(2):24, August 2011. 43, 109

[8] Florian Rodler. *CRIRES User Manual*. ESO, http://www.eso.org/sci/facilities/paranal/instruments/crires/doc.html, 3.1 edition, 2021. ESO-254264. 10, 12, 29

[9] L. F. Sarmiento, A. Reiners, P. Huke, F. F. Bauer, E. W. Guenter, U. Seemann, and U. Wolter. Comparing the emission spectra of U and Th hollow cathode lamps and a new U line list. *Astronomy and Astrophysics*, 618:A118, October 2018. 43, 109

[10] A. Smette, H. Sana, S. Noll, H. Horst, W. Kausch, S. Kimeswenger, M. Barden, C. Szyszka, A. M. Jones, A. Gallenne, J. Vinther, P. Ballester, and J. Taylor. Molecfit: A general tool for telluric absorption correction. I. Method and application to ESO instruments. *Astronomy and Astrophysics*, 576:A77, April 2015. 107

[11] F. Stoehr, R. White, M. Smith, I. Kamp, R. Thompson, D. Durand, W. Freudling, D. Fraquelli, J. Haase, R. Hook, T. Kimball, M. Kümmel, K. Levay, M. Lombardi, A. Micol, and T. Rogers. DER_SNR: A Simple & General Spectroscopic Signal-to-Noise Measurement Algorithm. In R. W. Argyle, P. S. Bunclark, and J. R. Lewis, editors, *Astronomical Data Analysis Software and Systems XVII*, volume 394 of *Astronomical Society of the Pacific Conference Series*, page 505, August 2008. 41