APERO Documentation

Version 0.6.131

Neil Cook

Last updated : September, 2020

Table of contents

Chapter 1

Latest version: 0.6.131

APERO is a pipeline designed to reduce astrophysical observations (specifically from echelle spectrographs). It is the official pipeline for:

- SPIROU (SPectropolarimeter InfraROUge) on the Canada-France-Hawaii Telescope CFHT.
- NIRPS (Currently under construction)

1.1 General User documentation

This section provides a general guide to using APERO.

1.1.1 Installation

Once you have installed APERO you can read about running APERO *here*.

1.1.1.1 Prerequisites

APERO is coded using python 3 Please do not use python 2 with APERO.

The following python modules are required

```
astropy
matplotlib
pandas
numpy
scipy
yaml
```

The following python modules are recommended

```
astroquery (required for precision BERV)
barycorrpy (required for precision BERV)
bottleneck (faster)
ipdb       (debugging)
numba      (faster)
PIL        (some graphical interfaces)
tqdm       (for some tools)
yagmail    (for an email settings)
```

See *here* for guide to intalling python and modules with the recommended setup.

1.1.1.2 Download from GitHub

Change to your desired installation directory (from now on this is referred to as *DRS_ROOT*) e.g. `/home/user/bin/apero-drs/`

Clone

Clone from github

```
git clone https://github.com/njcuk9999/apero-drs
```

This may take some time (in future most of the data required will be a separate download), and we still have many (now redundant) files from the spirou_py3 repository.

Choose branch

Change to the *{DRS_ROOT}* directory

Choose which branch:

- **master version** This is the version currently recommended for all general use. It may not contain the most up-to-date features until long term support and stability can be verified.
  Change to this branch with

  ```
  git checkout master
  git pull origin master
  ```

- **developer version** Note the developer version should have been tested and semi-stable but not ready for full sets of processing and definitely not for release for non-developers or for data put on archives. Some changes may not be in this version that are in the working version.
  Change to this branch with

  ```
  git checkout developer
  git pull origin developer
  ```

- **working version** Note the working version will be the most up-to-date version but has not been tested for stability - use at own risk.
  Change to this branch with

  ```
  git checkout working
  git pull origin working
  ```

1.1.1.3 Setup

Run the installation script

Change to the *{DRS_ROOT}* directory

Run the installation script

```
python setup/install.py --name={PROFILE}
```

where {*PROFILE*} is a short descriptive name for a setup (you can have multiple profiles with one installation)

e.g.

```
python setup/install.py --name=setup_njc_200903
```

Step-by-step guide

Follow the step-by-step guide:

- A: User configuration path
  This is the path where your configuration will be saved. If it doesn't exist you will be prompted to create it. (This will be referred to as *DRS_UCONFIG* from now on (default is `/home/user/apero/`*`PROFILE`*)
- B: Instrument settings
  Install *INSTRUMENT*. If yes it will install the instrument if not then it will not install the instrument. Currently only SPIRou is supported
- C: Set up paths
  The first question will ask whether to set up paths individually. If *[Y]es* it will allow you to set each path separately (i.e. for raw, tmp, reduced, calibDB etc). If *[N]o* you will just set one path and all folders (raw, tmp, reduced, calibDB etc)) will be created under this directory.
- D: Setting the directory/directories
  Will prompt you to enter the directory path/paths (will ask you for each if you answered that paths be set up individually in step C above.
- E: Clean install
  If you type [Y]es you will be prompted (later) to reset the directories this means any previous data in these directories will be removed. Note you can always say later to individual cases.

Warning: Resetting a directory will remove all files/sub-directories from within these folders

Note: A to E will repeat for all installable instruments (To step up just one use the *–instrument* argument

Additional options

One will be prompted to give installation paths to various optional tools (currently *ds9* and *pdflatex* note the user will not be prompted if these were automatically found using the *where* command)

1.1.1.4 Activating the APERO profile

To activate an apero profile you need to source the *{DRS_UCONGIG}/{PROFILE}.{SYSTEM}*.setup script.

Details of this should be in green at the end of the installation process

i.e. for bash:

```
source {DRS_UCONFIG}/{PROFILE}.bash.setup
```

i.e. for tcsh/csh/sh

```
source {DRS_UCONFIG}/{PROFILE}.sh.setup
```

e.g. with bash and our example profile above:

```
source {DRS_UCONFIG}/{PROFILE}.sh.setup
```

We strongly recommend setting up a alias for this

i.e. for bash (i.e. in `~/.bashrc` `~/.profile` or `~/.bash_aliases`):

```
alias {PROFILE}="source {DRS_UCONFIG}/{PROFILE}.bash.setup"
```

i.e. for tcsh/csh/sh (i.e. in `~/.tcshrc`, `~/.cshrc` etc)

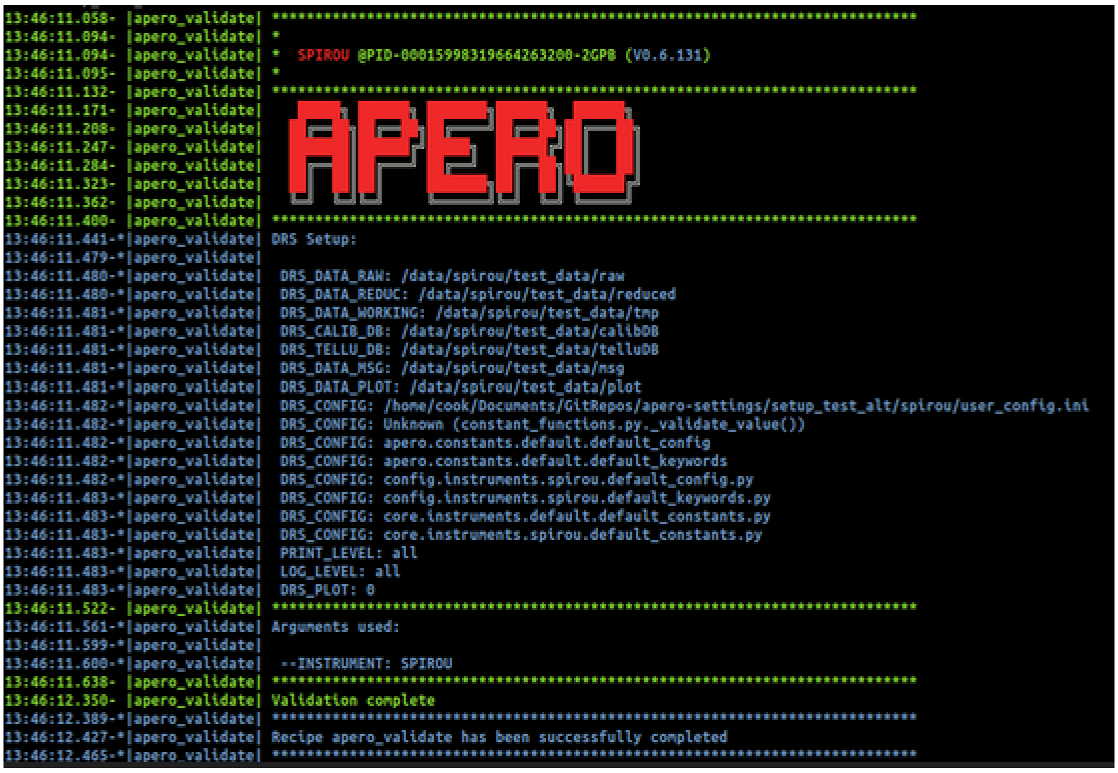
```
alias {PROFILE} "source {DRS_UCONFIG}/{PROFILE}.sh.setup"
```

Note: This must be done every time one wishes to use APERO (and must be done after one activates the conda environment

conda activate apero-env

One could add these both to automatically happen in a `~/.bashrc` but we recommend activating each time.

Following on from typing this command you should see a splash screen validating the installation and letting you know everything is good to run APERO recipes and tools.



```
13:46:11.058- |apero_validate| ********************************************************************************
13:46:11.094- |apero_validate| *
13:46:11.094- |apero_validate| *  SPIROU @PID-00015998319664263200-2GPB (V0.6.131)
13:46:11.095- |apero_validate| *
13:46:11.132- |apero_validate| ********************************************************************************
13:46:11.171- |apero_validate|
13:46:11.208- |apero_validate|
13:46:11.247- |apero_validate|
13:46:11.284- |apero_validate|
13:46:11.323- |apero_validate|
13:46:11.362- |apero_validate|
13:46:11.400- |apero_validate| ********************************************************************************
13:46:11.441-*|apero_validate| DRS Setup:
13:46:11.479-*|apero_validate|
13:46:11.480-*|apero_validate|  DRS_DATA_RAW: /data/spirou/test_data/raw
13:46:11.480-*|apero_validate|  DRS_DATA_REDUC: /data/spirou/test_data/reduced
13:46:11.481-*|apero_validate|  DRS_DATA_WORKING: /data/spirou/test_data/tmp
13:46:11.481-*|apero_validate|  DRS_CALIB_DB: /data/spirou/test_data/calibDB
13:46:11.481-*|apero_validate|  DRS_TELLU_DB: /data/spirou/test_data/telluDB
13:46:11.481-*|apero_validate|  DRS_DATA_MSG: /data/spirou/test_data/msg
13:46:11.481-*|apero_validate|  DRS_DATA_PLOT: /data/spirou/test_data/plot
13:46:11.482-*|apero_validate|  DRS_CONFIG: /home/cook/Documents/GitRepos/apero-settings/setup_test_alt/spirou/user_config.ini
13:46:11.482-*|apero_validate|  DRS_CONFIG: Unknown (constant_functions.py._validate_value())
13:46:11.482-*|apero_validate|  DRS_CONFIG: apero.constants.default.default_config
13:46:11.482-*|apero_validate|  DRS_CONFIG: apero.constants.default.default_keywords
13:46:11.482-*|apero_validate|  DRS_CONFIG: config.instruments.spirou.default_config.py
13:46:11.483-*|apero_validate|  DRS_CONFIG: config.instruments.spirou.default_keywords.py
13:46:11.483-*|apero_validate|  DRS_CONFIG: core.instruments.default.default_constants.py
13:46:11.483-*|apero_validate|  DRS_CONFIG: core.instruments.spirou.default_constants.py
13:46:11.483-*|apero_validate|  PRINT_LEVEL: all
13:46:11.483-*|apero_validate|  LOG_LEVEL: all
13:46:11.483-*|apero_validate|  DRS_PLOT: 0
13:46:11.522- |apero_validate| ********************************************************************************
13:46:11.561-*|apero_validate| Arguments used:
13:46:11.599-*|apero_validate|
13:46:11.600-*|apero_validate|  --INSTRUMENT: SPIROU
13:46:11.638- |apero_validate| ********************************************************************************
13:46:12.350- |apero_validate| Validation complete
13:46:12.389-*|apero_validate| ********************************************************************************
13:46:12.427-*|apero_validate| Recipe apero_validate has been successfully completed
13:46:12.465-*|apero_validate| ********************************************************************************
```

1.1.1.5 Updating from github

1. Choose a branch (as in *Choose branch*)
2. Update the branch (pull from github)

   ```
   git pull origin {branch}
   ```

3. Make sure you are in an APERO profile

   ```
   source {DRS_UCONFIG}/{PROFILE}/setup.bash.setup
   ```

   or if you have it aliased

   ```
   {PROFILE}
   ```

   e.g.

   ```
   source /home/user/apero/setup_njc_200903/setup.bash.setup
   ```

   or if you have it aliased

   ```
   setup_njc_200903
   ```

4. Update using the installation script

```
python setup/install.py --update
```

This will use all current settings and update the

1.1.2 Using APERO

The user scripts to reduce data are referred to as 'recipes'.

From a coding point of view this due to the fact that they literally list the steps (where each step is a function or set of functions).

By design recipes are kept to a bare minimum of code and all heavy functionality is done in the functions that are called in the recipes.

Currently supported instruments are:

- SPIRou (See the section on recipes *here*)

There are two ways to use APERO:

1. Using recipes individually
2. Using the processing script to automatically generate batches of recipe runs (based on provided run files)

both of these require installation (see *here*) and activating a profile (see the next section *here*)

1.1.2.1 Activating the APERO profile

To activate an apero profile you need to source the *{DRS_UCONGIG}/{PROFILE}.{SYSTEM}*.setup script.

Details of this should be in green at the end of the installation process

i.e. for bash:

```
source {DRS_UCONFIG}/{PROFILE}.bash.setup
```

i.e. for tcsh/csh/sh

```
source {DRS_UCONFIG}/{PROFILE}.sh.setup
```

e.g. with bash and our example profile above:

```
source {DRS_UCONFIG}/{PROFILE}.sh.setup
```

We strongly recommend setting up a alias for this

i.e. for bash (i.e. in `~/.bashrc` `~/.profile` or `~/.bash_aliases`):

```
alias {PROFILE}="source {DRS_UCONFIG}/{PROFILE}.bash.setup"
```

i.e. for tcsh/csh/sh (i.e. in `~/.tcshrc`, `~/.cshrc` etc)

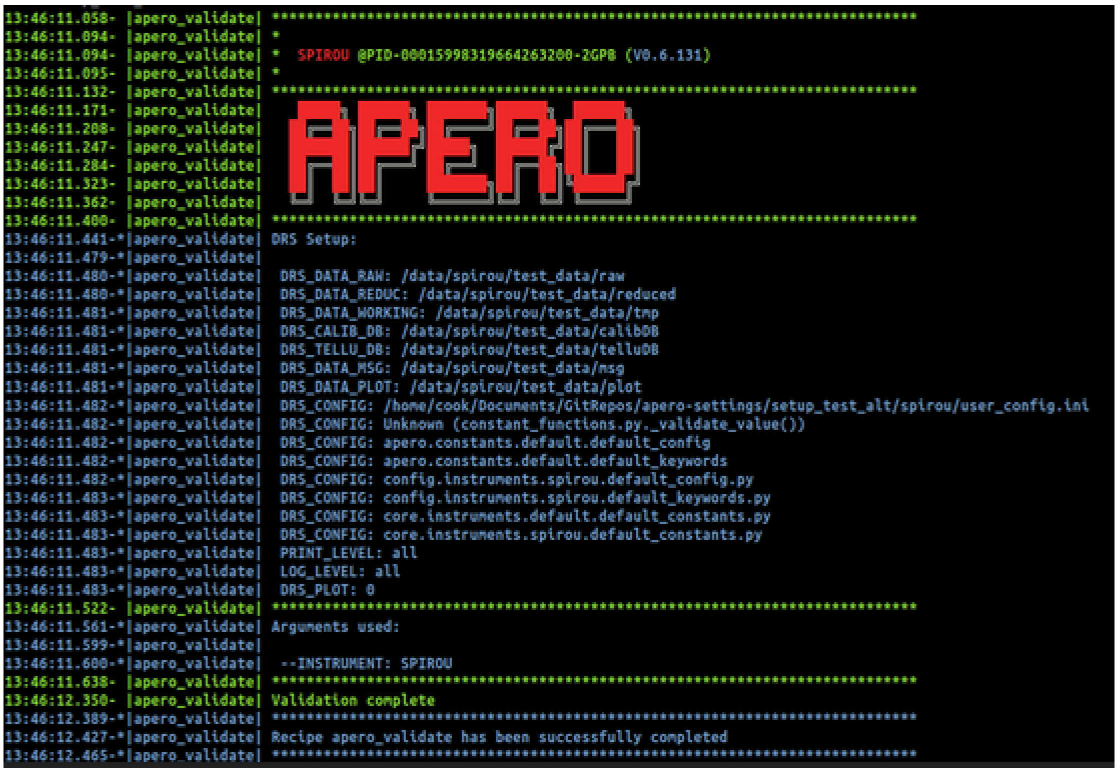
```
alias {PROFILE} "source {DRS_UCONFIG}/{PROFILE}.sh.setup"
```

Note: This must be done every time one wishes to use APERO (and must be done after one activates the conda environment

conda activate apero-env

One could add these both to automatically happen in a `~/.bashrc` but we recommend activating each time.

Following on from typing this command you should see a splash screen validating the installation and letting you know everything is good to run APERO recipes and tools.



```
13:46:11.058- |apero_validate| ***************************************************************************
13:46:11.094- |apero_validate| *
13:46:11.094- |apero_validate| *  SPIROU @PID-00015998319664263200-2GPB (V0.6.131)
13:46:11.095- |apero_validate| *
13:46:11.132- |apero_validate| ***************************************************************************
13:46:11.171- |apero_validate|
13:46:11.208- |apero_validate|
13:46:11.247- |apero_validate|
13:46:11.284- |apero_validate|
13:46:11.323- |apero_validate|
13:46:11.362- |apero_validate|
13:46:11.400- |apero_validate| ***************************************************************************
13:46:11.441-*|apero_validate| DRS Setup:
13:46:11.479-*|apero_validate|
13:46:11.480-*|apero_validate|  DRS_DATA_RAW: /data/spirou/test_data/raw
13:46:11.480-*|apero_validate|  DRS_DATA_REDUC: /data/spirou/test_data/reduced
13:46:11.481-*|apero_validate|  DRS_DATA_WORKING: /data/spirou/test_data/tmp
13:46:11.481-*|apero_validate|  DRS_CALIB_DB: /data/spirou/test_data/calibDB
13:46:11.481-*|apero_validate|  DRS_TELLU_DB: /data/spirou/test_data/telluDB
13:46:11.481-*|apero_validate|  DRS_DATA_MSG: /data/spirou/test_data/msg
13:46:11.481-*|apero_validate|  DRS_DATA_PLOT: /data/spirou/test_data/plot
13:46:11.482-*|apero_validate|  DRS_CONFIG: /home/cook/Documents/GitRepos/apero-settings/setup_test_alt/spirou/user_config.ini
13:46:11.482-*|apero_validate|  DRS_CONFIG: Unknown (constant_functions.py._validate_value())
13:46:11.482-*|apero_validate|  DRS_CONFIG: apero.constants.default.default_config
13:46:11.482-*|apero_validate|  DRS_CONFIG: apero.constants.default.default_keywords
13:46:11.482-*|apero_validate|  DRS_CONFIG: config.instruments.spirou.default_config.py
13:46:11.483-*|apero_validate|  DRS_CONFIG: config.instruments.spirou.default_keywords.py
13:46:11.483-*|apero_validate|  DRS_CONFIG: core.instruments.default.default_constants.py
13:46:11.483-*|apero_validate|  DRS_CONFIG: core.instruments.spirou.default_constants.py
13:46:11.483-*|apero_validate|  PRINT_LEVEL: all
13:46:11.483-*|apero_validate|  LOG_LEVEL: all
13:46:11.483-*|apero_validate|  DRS_PLOT: 0
13:46:11.522- |apero_validate| ***************************************************************************
13:46:11.561-*|apero_validate| Arguments used:
13:46:11.599-*|apero_validate|
13:46:11.600-*|apero_validate|  --INSTRUMENT: SPIROU
13:46:11.638- |apero_validate| ***************************************************************************
13:46:12.350- |apero_validate| Validation complete
13:46:12.389-*|apero_validate| ***************************************************************************
13:46:12.427-*|apero_validate| Recipe apero_validate has been successfully completed
13:46:12.465-*|apero_validate| ***************************************************************************
```

1.1.2.2 Running recipes indvidiually

How to run an individual recipe.

1.1.2.3 Using the processing script

How to use the APERO processing script.

1.1.3 Known Issues

Currently known issues and problems with APERO. Last updated: 2020-07-24 (NJC).

1.1.3.1 Recipes

- Locking of files means using >5 cores slows down a lot (fix with database)
- Weird residuals left in order_profile after dark_flat (loc)
- Calibrations switch over at different points from PM to AM calibrations (should really only use "older")
- NIRPS breaks in shape

1.1.3.2 External

- can't use barycorrpy –> update astropy (version 4.0) and barycorrpy (version >0.3))

1.1.4 TODO

This is the currently list of items that need to still be completed. Last updated: 2020-05-04 (NJC).

Note: bullet points are not ordered

1.1.4.1 APERO

For all instruments / in general.

High priority:

- fix headers for having bad wave and loc keys (remove?) [NEEDS DISCUSSION]

Low priority:

- go through all summary plots and decide which plots, write figure captions, improve plots, write quality control description, decide which header keys to print
- add all functions doc strings
- write documentation
  - code to write constants/keywords
  - write doc strings
  - autodoc with sphinx once doc strings are in
  - assign people to write constant descriptions
  - add authors to constants
- newprofile.py outpaths for setup files are not correct (missing directory)
- add more debug printouts
- deal with all python warnings
- display func for all functions
- add doc strings to all functions, descriptions to all constants, review all constant min/max/dtypes
- proper SNR calculation
- add option to set template for mk_tellu and fit_tellu
- add flag for parellel / non parellel mode - can disable locking then?
- named break points
- add trigger option to drs where processing script stops when it cannot get any further with current set of files
- data separate download from DRS
  - this includes copying this data to the user data directory for adding things like custom masks etc - this could also be a solution to where to put the *object_query_list.fits* file

Later:

- CCF wrapper for weighting?
- persistence correction
- add EA mask generation from templates
- add EA template matching
- uncertainty propagation
- add *plot== 3* (all debug plots shown) and *plot==4* (all debug plots saved) modes
- move *object_query_list.fits* to *calibDB*
- co-production of e2ds and e2dsff still needed?
- write paper
- setup instrument tool
- Windows compatibility
- add *cal_drift* recipe?

1.1.4.2 SPIRou specific

High priority:

- bisector for CCF (new extension in CCF outputs?)
- object database (in preprocessing?)
- polar code update
- EA masks from templates

Low priority:

- output files like CFHT (e.fits, p.fits, v.fits etc)
- finish *obj_spec_spirou* and *obj_pol_spirou* [Do not use them now]

1.1.4.3 NIRPS specific

High priority:

- convert/adapt cal_preprocessing
- convert/adapt cal_wave / cal_wave_master

Low priority:

- convert obj_mk_tellu
- convert obj_fit_tellu
- convert obj_mk_template
- convert cal_ccf

Later:

- T.B.D.

1.1.5 General Tools

For instrument specific guide see:

- *SPIROU*

1.2 SPIRou documentation

1.2.1 Using APERO with SPIRou

This section describes the process to use APERO.

1.2.2 SPIRou Recipes

This section describes are all the SPIROU recipes to use with APERO.

For information on how to run these recipes (either individually or with the processing tools) see *here*.

1.2.2.1 Preprocessing Recipe

Cleans file of detector effects.

Run

```
cal_preprocess_spirou.py [DIRECTORY] [RAW_FILES]
```

Optional Arguments

```
--skip, --debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir:

```
DRS_DATA_WORKING   \\ default: "tmp" directory
```

Output files

```
{ODOMETER_CODE}_pp.fits  \\ preprocessed files (4096x4096)
```

Plots

```
None
```

1.2.2.2 Dark Master Recipe

Collects all dark files and creates a master dark image to use for correction.

Run

```
cal_dark_master_spirou.py
```

Optional Arguments

```
--filetype, --database, --plot,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
DARKM {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_dark_master.fits  \\ dark master file (4096x4096) + FITS-TABLE
```

Plots

```
None
```

Notes

Does not require a master night choice - finds darks from all preprocessed nights.

1.2.2.3 Bad Pixel Correction Recipe

Creates a bad pixel mask for identifying and deal with bad pixels.

Run

```
cal_badpix_spirou.py [DIRECTORY] -flatfiles [FLAT_FLAT] -darkfiles [DARK_DARK_TEL]
cal_badpix_spirou.py [DIRECTORY] -flatfiles [FLAT_FLAT] -darkfiles [DARK_DARK_INT]
```

Optional Arguments

```
--database, --combine, --flipimage, --fluxunits, --plot, --resize,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
BADPIX {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
BKGRDMAP {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_badpixel.fits  \\ bad pixel map file (3100x4088)
{ODOMETER_CODE}_pp_bmap.fits      \\ background mask file (3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits \\ debug background file (7x3100x4088)
```

Plots

```
BADPIX_MAP
```

1.2.2.4 Localisation Recipe

Finds the orders on the image.

Run

```
cal_loc_spirou.py [DIRECTORY] [FLAT_DARK]
cal_loc_spirou.py [DIRECTORY] [DARK_FLAT]
```

Optional Arguments

```
--database, --badpixfile, --badcorr, --backsub, --combine,
--darkfile, --darkcorr,  --flipimage, --fluxunits, --plot, --resize,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
ORDER_PROFILE_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
LOC_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_order_profile_C.fits  \\ order profile file (3100x4088)
{ODOMETER_CODE}_pp_loco_C.fits           \\ localisation centers map file (49x4088)
{ODOMETER_CODE}_pp_fwhm-order_C.fits     \\ localisation widths map file (49x4088)
{ODOMETER_CODE}_pp_with-order_C.fits     \\ localisation superposition file (3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits \\ debug background file (7x3100x4088)
```

Plots

```
LOC_MINMAX_CENTS, LOC_MIN_CENTS_THRES, LOC_FINDING_ORDERS, LOC_IM_SAT_THRES,
LOC_ORD_VS_RMS, LOC_CHECK_COEFFS, LOC_FIT_RESIDUALS
```

1.2.2.5 Shape Master Recipe

Creates a master FP image from all FPs processed. Uses this to work out the required shifts due to the FP master image, slicer pupil geometry and the bending of the orders (found in localisation).

Run

```
cal_shape_master_spirou.py [DIRECTORY] -hcfiles [HCONE_HCONE] -fpfiles [FP_FP]
```

Optional Arguments

```
--database, --badpixfile, --badcorr, --backsub, --combine,
--darkfile, --darkcorr,  --flipimage, --fluxunits, --locofile,
--plot, --resize,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
SHAPEX {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
SHAPEY {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
FPMASTER {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_shapex.fits            \\ dx shape map (3100x4088)
{ODOMETER_CODE}_pp_shapey.fits            \\ dy shape map (3100x4088)
{ODOMETER_CODE}_pp_fpmaster.fits          \\ fp master file (3100x4088) + FITS-TABLE
DEBUG_{ODOMETER_CODE}_shape_out_bdx.fits  \\ dx map before dy map (3100x4088)
DEBUG_{ODOMETER_CODE}_shape_in_fp.fits    \\ input fp before shape corr (3100x4088)
DEBUG_{ODOMETER_CODE}_shape_out_fp.fits   \\ input fp after shape corr (3100x4088)
DEBUG_{ODOMETER_CODE}_shape_in_hc.fits    \\ input hc before shape corr (3100x4088)
DEBUG_{ODOMETER_CODE}_shape_out_hc.fits   \\ input hc after shape corr (3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits \\ debug background file (7x3100x4088)
```

Plots

```
SHAPE_DX, SHAPE_ANGLE_OFFSET_ALL, SHAPE_ANGLE_OFFSET, SHAPE_LINEAR_TPARAMS
```

1.2.2.6 Shape (per night) Recipe

Takes the shape master outputs (shapex, shapey and fpmaster) and applies these transformations to shift the image to the master fp frame, unbend images and shift to correct for slicer pupil geometry.

Run

```
cal_shape_spirou.py [DIRECTORY] [FP_FP]
```

Optional Arguments

```
--database, --badpixfile, --badcorr, --backsub, --combine,
--darkfile, --darkcorr,  --flipimage, --fluxunits, --fpmaster,
--plot, --resize, --shapex, --shapey,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
SHAPEL {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_shapel.fits            \\ local shape map (3100x4088)
DEBUG_{ODOMETER_CODE}_shape_in_fp.fits    \\ input fp before shape corr (3100x4088)
DEBUG_{ODOMETER_CODE}_shape_out_fp.fits   \\ input fp after shape corr (3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits \\ debug background file (7x3100x4088)
```

Plots

```
SHAPE_DX, SHAPE_ANGLE_OFFSET_ALL, SHAPE_ANGLE_OFFSET, SHAPE_LINEAR_TPARAMS
```

1.2.2.7 Flat/Blaze Correction Recipe

Extracts out flat images in order to measure the blaze and produced blaze correction and flat correction images.

Run

```
cal_flat_spirou.py [DIRECTORY] [FLAT_FLAT]
```

Optional Arguments

```
--database, --badpixfile, --badcorr, --backsub, --combine,
--darkfile, --darkcorr,  --fiber, --flipimage, --fluxunits,
--locofile, --orderpfile, --plot, --resize,
--shapex, --shapey, --shapel,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
FLAT_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
BLAZE_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_blaze_{FIBER}.fits         \\ blaze correction file (49x4088)
{ODOMETER_CODE}_pp_flat_{FIBER}.fits          \\ blaze correction file (49x4088)
DEBUG_{ODOMETER_CODE}_pp_e2dsll_{FIBER}.fits  \\ debug pre extract file (7x3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits      \\ debug background file (7x3100x4088)
```

Plots

```
FLAT_ORDER_FIT_EDGES1, FLAT_ORDER_FIT_EDGES2, FLAT_BLAZE_ORDER1,
FLAT_BLAZE_ORDER2
```

1.2.2.8 Thermal Correction Recipe

Extracts dark frames in order to provide correction for the thermal background after extraction of science / calibration frames.

Run

```
cal_thermal_spirou.py [DIRECTORY] [DARK_DARK_INT]
cal_thermal_spirou.py [DIRECTORY] [DARK_DARK_TEL]
```

Optional Arguments

```
--database, --badpixfile, --badcorr, --backsub, --combine,
--darkfile, --darkcorr,  --fiber, --flipimage, --fluxunits,
--locofile, --orderpfile, --plot, --resize,
--shapex, --shapey, --shapel, --wavefile,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
THERMALT_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
THERMALI_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_e2ds_{FIBER}.fits              \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_e2dsff_{FIBER}.fits            \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_s1d_w_{FIBER}.fits             \\ s1d constant in pixel space (FITS-TABLE)
{ODOMETER_CODE}_pp_s1d_v_{FIBER}.fits             \\ s1d constant in velocity space (FITS-TABLE)
DEBUG_{ODOMETER_CODE}_pp_e2dsll_{FIBER}.fits      \\ debug pre extract file (7x3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits       \\ debug background file (7x3100x4088)
{ODOMETER_CODE}_pp_thermal_e2ds_int_{FIBER}.fits  \\ extracted thermal for dark_dark_int (49x4088)
{ODOMETER_CODE}_pp_thermal_e2ds_tel_{FIBER}.fits  \\ extracted thermal for dark_dark_tel (49x4088)
```

Plots

```
None
```

1.2.2.9 Master leak correction recipe

Extracts all DARK_FP files and creates a model for later leak correction.

Run

```
cal_leak_master.py [DIRECTORY]
```

Optional Arguments

```
--filetype, --database, --plot
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
WAVE_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_e2dsff_{FIBER}.fits            \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_leak_master_{FIBER}.fits        \\ leak correction maste rfile (49x4088)
```

Plots

```
None
```

1.2.2.10 Master wavelength solution Recipe

Creates a wavelength solution and measures drifts (via CCF) of the FP relative to the FP master

Run

```
cal_wave_master_spirou.py [DIRECTORY] -hcfiles [HCONE_HCONE] -fpfiles [FP_FP]
```

Optional Arguments

```
--database, --badpixfile, --badcorr, --backsub, --blazefile,
--combine, --darkfile, --darkcorr,  --fiber, --flipimage,
--fluxunits,  --locofile, --orderpfile, --plot, --resize,
--shapex, --shapey, --shapel, --wavefile, -hcmode, -fpmode,
--forceext,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
WAVEM_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
WAVEHCL_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
WAVEFPL_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_e2ds_{FIBER}.fits            \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_e2dsff_{FIBER}.fits          \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_s1d_w_{FIBER}.fits           \\ s1d constant in pixel space (FITS-TABLE)
{ODOMETER_CODE}_pp_s1d_v_{FIBER}.fits           \\ s1d constant in velocity space (FITS-TABLE)
DEBUG_{ODOMETER_CODE}_pp_e2dsll_{FIBER}.fits    \\ debug pre extract file (7x3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits        \\ debug background file (7x3100x4088)

{ODOMETER_CODE}_pp_e2dsff_linelist_{FIBER}.dat    \\ wave stats hc line list
{ODOMETER_CODE}_pp_e2dsff_wavemres_{FIBER}.fits   \\ wave res table (multi extension fits)
{ODOMETER_CODE}_pp_e2dsff_wavem_hc_{FIBER}.fits   \\ wave solution from hc only (49x4088)
{ODOMETER_CODE}_pp_e2dsff_wavem_fp_{FIBER}.fits   \\ wave solution from hc + fp (49x4088)
cal_wave_results.tbl                              \\ wave res table (ASCII-table)
{ODOMETER_CODE}_pp_e2dsff_mhc_lines_{FIBER}.tbl   \\ wave hc lines (ASCII-table)
{ODOMETER_CODE}_pp_wavem_hclines_{FIBER}.fits     \\ wave hc ref/measured lines table (FITS-TABLE)
{ODOMETER_CODE}_pp_wavem_fplines_{FIBER}.fits    \\ wave fp ref/measured lines table (FITS-TABLE)
{ODOMETER_CODE}_pp_e2dsff_ccf_{FIBER}.fits        \\ ccf code [FITS-TABLE]
```

Plots

```
WAVE_HC_GUESS, WAVE_HC_BRIGHTEST_LINES, WAVE_HC_TFIT_GRID, WAVE_HC_RESMAP, WAVE_LITTROW_CHECK1,
WAVE_LITTROW_EXTRAP1, WAVE_LITTROW_CHECK2, WAVE_LITTROW_EXTRAP2, WAVE_FP_FINAL_ORDER,
WAVE_FP_LWID_OFFSET, WAVE_FP_WAVE_RES, WAVE_FP_M_X_RES, WAVE_FP_IPT_CWID_1MHC, WAVE_FP_IPT_CWID_LLHC,
WAVE_FP_LL_DIFF, WAVE_FP_MULTI_ORDER, WAVE_FP_SINGLE_ORDER, CCF_RV_FIT, CCF_RV_FIT_LOOP, WAVEREF_
↪EXPECTED,
EXTRACT_S1D, EXTRACT_S1D_WEIGHT, WAVE_FIBER_COMPARISON, WAVE_FIBER_COMP, WAVENIGHT_ITERPLOT, WAVENIGHT_
↪HISTPLOT
```

1.2.2.11 Nightly wavelength solution Recipe

Calculates corrections to the master wavelength solution as a nightly wavelength solution and measures drifts (via CCF) of the FP relative to the FP master

Run

```
cal_wave_night_spirou.py [DIRECTORY] -hcfiles [HCONE_HCONE] -fpfiles [FP_FP]
```

Optional Arguments

```
--database, --badpixfile, --badcorr, --backsub, --blazefile,
--combine, --darkfile, --darkcorr,  --fiber, --flipimage,
--fluxunits,  --locofile, --orderpfile, --plot, --resize,
--shapex, --shapey, --shapel, --wavefile, -hcmode, -fpmode,
--forceext
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Calibration database entry

```
WAVE_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_e2ds_{FIBER}.fits              \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_e2dsff_{FIBER}.fits            \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_s1d_w_{FIBER}.fits             \\ s1d constant in pixel space (FITS-TABLE)
{ODOMETER_CODE}_pp_s1d_v_{FIBER}.fits             \\ s1d constant in velocity space (FITS-TABLE)
DEBUG_{ODOMETER_CODE}_pp_e2dsll_{FIBER}.fits      \\ debug pre extract file (7x3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits          \\ debug background file (7x3100x4088)

{ODOMETER_CODE}_pp_e2dsff__wave_night_{FIBER}.fits     \\ wave night solution (49x4088)
{ODOMETER_CODE}_pp_wavem_hclines_{FIBER}.fits      \\ wave hc ref/measured lines table (FITS-TABLE)
{ODOMETER_CODE}_pp_wavem_fplines_{FIBER}.fits     \\ wave fp ref/measured lines table (FITS-TABLE)
{ODOMETER_CODE}_pp_e2dsff_ccf_{FIBER}.fits         \\ ccf code [FITS-TABLE]
```

Plots

```
WAVENIGHT_ITERPLOT WAVENIGHT_HISTPLOT WAVEREF_EXPECTED
CCF_RV_FIT CCF_RV_FIT_LOOP EXTRACT_S1D EXTRACT_S1D_WEIGHT
```

1.2.2.12 Extraction Recipe

Extracts any preprocessed image using all the calibrations required.

Run

```
cal_extract_spirou.py [DIRECTORY] [PP_FILE]
```

Optional Arguments

```
--badpixfile, --badcorr, --backsub, --blazefile,
--combine, --objname, --dprtype, --darkfile, --darkcorr,
--fiber, --flipimage, --fluxunits, --flatfile,
--locofile, --orderpfile, --plot, --resize,
--shapex, --shapey, --shapel, --thermal, --thermalfile, --wavefile,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Output files

```
{ODOMETER_CODE}_pp_e2ds_{FIBER}.fits           \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_e2dsff_{FIBER}.fits         \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_s1d_w_{FIBER}.fits          \\ s1d constant in pixel space (FITS-TABLE)
{ODOMETER_CODE}_pp_s1d_v_{FIBER}.fits          \\ s1d constant in velocity space (FITS-TABLE)
DEBUG_{ODOMETER_CODE}_pp_e2dsll_{FIBER}.fits   \\ debug pre extract file (7x3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits       \\ debug background file (7x3100x4088)
{ODOMETER_CODE}_pp_ext_fplines_{FIBER} .fits   \\ the FP ref/measured lines (FOR OBJ_FP only)
```

Plots

```
FLAT_ORDER_FIT_EDGES1, FLAT_ORDER_FIT_EDGES2, FLAT_BLAZE_ORDER1,
FLAT_BLAZE_ORDER2, THERMAL_BACKGROUND, EXTRACT_SPECTRAL_ORDER1,
EXTRACT_SPECTRAL_ORDER2, EXTRACT_S1D, EXTRACT_S1D_WEIGHT
```

1.2.2.13 Leak correction Recipe

Corrects extracted files for leakage coming from a FP (for OBJ_FP files only)

Run

```
cal_leak_spirou.py [DIRECTORY] [PP_FILE]
```

Optional Arguments

```
--database, --plot, --leakfile
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Output files

```
{ODOMETER_CODE}_pp_e2ds_{FIBER}.fits              \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_e2dsff_{FIBER}.fits            \\ extracted + flat field file (49x4088)
{ODOMETER_CODE}_pp_s1d_w_{FIBER}.fits             \\ s1d constant in pixel space (FITS-TABLE)
{ODOMETER_CODE}_pp_s1d_v_{FIBER}.fits             \\ s1d constant in velocity space (FITS-TABLE)
DEBUG_{ODOMETER_CODE}_pp_e2dsll_{FIBER}.fits      \\ debug pre extract file (7x3100x4088)
DEBUG_{ODOMETER_CODE}_pp_background.fits          \\ debug background file (7x3100x4088)
```

Plots

```
None
```

1.2.2.14 Make Telluric Recipe

Takes a hot star and calculates telluric transmission

Run

```
obj_mk_tellu_spirou.py [DIRECTORY] [E2DS & OBJ_DARK]
obj_mk_tellu_spirou.py [DIRECTORY] [E2DSFF & OBJ_DARK]
obj_mk_tellu_spirou.py [DIRECTORY] [E2DS & OBJ_FP]
obj_mk_tellu_spirou.py [DIRECTORY] [E2DSFF & OBJ_FP]
```

Optional Arguments

```
--database, --blazefile, --plot, --wavefile
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Telluric database entry

```
TELLU_CONV_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
TELLU_TRANS_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_tellu_trans_{FIBER}.fits    \\ telluric transmission file (49x4088)
{WAVEFILE}_tellu_conv_{FIBER}.npy              \\ tapas convolved with wave file (49x4088)
```

Plots

```
MKTELLU_WAVE_FLUX1, MKTELLU_WAVE_FLUX2
```

1.2.2.15 Fit Telluric Recipe

Using the telluric tramission calculates principle components (PCA) to correct input images of atmospheric absorption.

Run

```
obj_fit_tellu_spirou.py [DIRECTORY] [E2DS & OBJ_DARK]
obj_fit_tellu_spirou.py [DIRECTORY] [E2DSFF & OBJ_DARK]
obj_fit_tellu_spirou.py [DIRECTORY] [E2DS & OBJ_FP]
obj_fit_tellu_spirou.py [DIRECTORY] [E2DSFF & OBJ_FP]
```

Optional Arguments

```
--database, --blazefile, --plot, --wavefile
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Telluric database entry

```
TELLU_CONV_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
TELLU_TRANS_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
{ODOMETER_CODE}_pp_e2dsff_tcorr_{FIBER}.fits    \\ telluric corrected e2dsff file (49x4088)
{ODOMETER_CODE}_pp_s1d_w_tcorr_{FIBER}.fits     \\ telluric corrected s1d constant in pixel space (FITS-
↪TABLE)
{ODOMETER_CODE}_pp_s1d_v_tcorr_{FIBER}.fits     \\ telluric corrected s1d constant in velocity space␣
↪(FITS-TABLE)
{ODOMETER_CODE}_pp_e2dsff_recon_{FIBER}.fits    \\ reconstructed transmission e2dsff file (49x4088)
{ODOMETER_CODE}_pp_s1d_w_recon_{FIBER}.fits     \\ reconstructed transmission s1d constant in pixel␣
↪space (FITS-TABLE)
{ODOMETER_CODE}_pp_s1d_v_recon_{FIBER}.fits     \\ reconstructed transmission s1d constant in velocity␣
↪space (FITS-TABLE)
```

Plots

```
EXTRACT_S1D, EXTRACT_S1D_WEIGHT, FTELLU_PCA_COMP1, FTELLU_PCA_COMP2,
FTELLU_RECON_SPLINE1, FTELLU_RECON_SPLINE2, FTELLU_WAVE_SHIFT1,
FTELLU_WAVE_SHIFT2, FTELLU_RECON_ABSO1, FTELLU_RECON_ABSO2
```

1.2.2.16 Make Template Recipe

Uses all telluric corrected images of a certain object name to create and BERV and wavelength corrected template in order to server as a better model SED for telluric correction.

obj_mk_tellu_spirou.py and *obj_fit_tellu_spirou.py* need to be rerun after template generation.

Run

```
obj_mk_template_spirou.py [OBJNAME]
```

Optional Arguments

```
--filetype, -fiber,
--database, --blazefile, --plot, --wavefile
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Telluric database entry

```
TELLU_CONV_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
TELLU_TRANS_{FIBER} {NIGHT_NAME} {FILENAME} {HUMAN DATE} {UNIX DATE}
```

Output files

```
Template_{OBJNAME}_{filetype}_{FIBER}.fits  \\ Template for object (3100x4088) + FITS TABLE
Template_s1d_{OBJNAME}_sc1d_w_{FIBER}.fits  \\ Template s1d constant in pixel space for object FITS␣
↪TABLE + FITS TABLE
Template_s1d_{OBJNAME}_sc1d_v_{FIBER}.fits  \\ Template s1d constant in velocity space for object FITS␣
↪TABLE + FITS TABLE
BigCube0_{OBJNAME}_{filetype}_{FIBER}.fits  \\ Cube of obs making template earth reference (49 x N x␣
↪4088)
BigCube_{OBJNAME}_{filetype}_{FIBER}.fits   \\ Cube of obs making template star reference (49 x N x 4088)
```

Plots

```
EXTRACT_S1D
```

1.2.2.17 CCF Recipe

Cross correlates the input image against a mask and measures a radial velocity per order, and combines to give an over all radial velocity measurement. Also (where possible) takes into account the FP drift measured by a CCF in the wave solution (when wave solution used a FP)

Run

```
cal_ccf_spirou.py [DIRECTORY] [E2DS & OBJ_FP]
cal_ccf_spirou.py [DIRECTORY] [E2DSFF & OBJ_FP]
cal_ccf_spirou.py [DIRECTORY] [E2DS_CORR & OBJ_FP]
cal_ccf_spirou.py [DIRECTORY] [E2DSFF_CORR & OBJ_FP]
cal_ccf_spirou.py [DIRECTORY] [E2DS & OBJ_DARK]
cal_ccf_spirou.py [DIRECTORY] [E2DSFF & OBJ_DARK]
cal_ccf_spirou.py [DIRECTORY] [E2DS_CORR & OBJ_DARK]
cal_ccf_spirou.py [DIRECTORY] [E2DSFF_CORR & OBJ_DARK]
```

Optional Arguments

```
--mask, --rv, --width, --step
--database, --blazefile, --plot
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Output files

```
{ODOMETER_CODE}_pp_{INTYPE}_{FIBER}_ccf_{mask}_{FIBER}.fits  \\ CCF for science channel file (FITS-TABLE)
{ODOMETER_CODE}_pp_{INTYPE}_{FIBER}_ccf_fp_{FIBER}.fits      \\ CCF for reference channel file (FITS-
↪TABLE)
```

Plots

```
CCF_RV_FIT, CCF_RV_FIT_LOOP, CCF_SWAVE_REF, CCF_PHOTON_UNCERT
```

1.2.2.18 Polarimetry Recipe

Produces all polarimetry outputs.

Run

```
pol_spirou.py [DIRECTORY] [E2DSFF]
pol_spirou.py [DIRECTORY] [E2DSFF_CORR]
```

Optional Arguments

```
--blazefile, --plot, --wavefile,
--debug, --listing, --listingall, --version, --info,
--program, --idebug, --breakpoints, --quiet, --help
```

Output Dir

```
DRS_DATA_REDUC   \\ default "reduced" directory
```

Output files

```
{ODOMETER_CODE}_pp_{INTYPE}_pol.fits             // polar file
{ODOMETER_CODE}_pp_{INTYPE}_StokesI.fits         // stokes file
{ODOMETER_CODE}_pp_{INTYPE}_null1_pol.fits       // null 1 file
{ODOMETER_CODE}_pp_{INTYPE}_null2_pol.fits       // null 2 file
{ODOMETER_CODE}_pp_{INTYPE}_lsd_pol.fits         // lsd file
{ODOMETER_CODE}_pp_{INTYPE}_s1d_w_pol.fits       // s1d polar file constant in pixel space for object
{ODOMETER_CODE}_pp_{INTYPE}_s1d_v_pol.fits       // s1d polar file constant in velocity space
{ODOMETER_CODE}_pp_{INTYPE}_s1d_w_null1.fits     // s1d null 1 file constant in pixel space for object
{ODOMETER_CODE}_pp_{INTYPE}_s1d_v_null1.fits     // s1d null 1 file constant in velocity space
{ODOMETER_CODE}_pp_{INTYPE}_s1d_w_null2.fits     // s1d null 2 file constant in pixel space for object
{ODOMETER_CODE}_pp_{INTYPE}_s1d_v_null2.fits     // s1d null 2 file constant in velocity space
{ODOMETER_CODE}_pp_{INTYPE}_s1d_w_stokesi.fits   // s1d stokes file constant in pixel space for object
{ODOMETER_CODE}_pp_{INTYPE}_s1d_v_stokesi.fits   // s1d stokes file constant in velocity space
```

Plots

```
POLAR_CONTINUUM, POLAR_RESULTS, POLAR_STOKES_I, POLAR_LSD,
EXTRACT_S1D, EXTRACT_S1D_WEIGHT
```

1.2.3 SPIRou Tools

These are all userful tools to use with APERO.

1.3 Developer how to guide

Below is a guide for those developing APERO for the current set of instruments and for future instruments.

1.3.1 Quick add links

1.3.1.1 Adding a new constant

1.3.1.2 Adding a new keyword

1.3.1.3 Adding a new recipe

1.3.1.4 Adding a new filetype

1.3.1.5 Adding a new plot

1.3.2 Full tutorials

- Const and Keyword
- ParamDict
- DrsRecipe, DrsArgument
- DrsInputFile, DrsFitsFile, DrsNpyFile
- Database
- Logger
- Debug Modes (linked to Logger)
- Plotter
- Git hub interface
- Writing documentation

1.3.3 Other

1.3.3.1 Required Input Header Keys

Main

```
OBSTYPE = 'OBJECT  '            / Observation / Exposure type ['DARK', 'FLAT', 'ALIGN', 'COMPARISON',
↪'OBJECT']
TRG_TYPE= 'TARGET  '            / target or sky object ['TARGET', 'SKY', '']
OBJECT  = 'HD159170'            / Target name
OBJNAME = 'HD159170'            / Target name   [DUPLICATE, should not be used any more use 'OBJECT']
```

object specific parameters

```
OBJRA    = '17:33:29.84'         / Target right ascension [HH:MM:SS.SSSS]
OBJDEC   = '-5:44:41.3'          / Target declination [DD:MM:SS.SSSS]
OBJEQUIN=               2000.0 / Target equinox
OBJRAPM =                 0.00 / Target right ascension proper motion in as/yr
OBJDECPM=                 0.00 / Target declination proper motion in as/yr
OBJTEMP =              9900.00 / Object effective temperature [K]
GAIA_ID =  1958476259155515008 / The Gaia ID used for BERV params  [OPTIONAL BUT RECOMMENDED]
OBJPLX  =     12.22203034418218 / PLX [mas] used to calc. BERV [OPTIONAL BUT RECOMMENDED]
OBSRV   =                   0.0 / RV [km/s] used to calc. BERV [OPTIONAL BUT RECOMMENDED]

AIRMASS =                1.151 / Airmass at start of observation
PI_NAME = 'QSO Team'
GAIN    =                0.999 / Amplifier gain (electrons/ADU)
RDNOISE =                 10.9 / Read noise (electrons)
FRMTIME =              5.57192 / [sec] Frame time, cadence of IR reads
EXPTIME =              300.884 / [sec] Integration time
SATURATE=                60000 / Saturation value (ADU)
```

Used to figure out what sequence files should be processed in:

```
CMPLTEXP=                    1 / Exposure number within the exposure sequence
NEXP    =                    1 / Total number of exposures within the sequence
```

Date:

```
MJDATE   =        58593.5428846 / Modified Julian Date at start of observation   [ONLY USED VISUALLY]
MJDEND   =        58593.5467937 / Modified Julian Date at end of observation
DATE-OBS= '2019-04-20'          / Date at start of observation (UTC) [YYYY-MM-DD]   [ONLY USED VISUALLY]
UTC-OBS = '13:01:45.23'         / Time at start of observation (UTC) [HH:MM:SS.SS]   [ONLY USED VISUALLY]
```

Used to know what position fibers/wheel where in:

```
SBCREF_P= 'pos_pk  '            / SPIRou Reference Fiber Position (predefined)
SBCCAS_P= 'pos_pk  '            / SPIRou Cassegrain Fiber Position (predefined)
SBCDEN_P=                 1.96 / SPIRou Calib-Reference density (0 to 3.3)  [ONLY USED FOR LOGGING?]
SBCALI_P= 'P5      '            / SPIRou calibwh predefined position or angle
```

Other

Used to report the air temperature [used in dark master – logging only?]:

```
TEMPERAT=                  2.33 / 86 temp, air, weather tower deg C
```

Used to report the cassigrain temperature [used in dark master – logging only?]:

```
SB_POL_T=                  4.03 / SPIRou tpolar temp at start of exp (deg C
```

used in dark master – logging only?:

```
RELHUMID=                  6.48 / 87 relative humidity, weather tower %
```

Chapter 2

Other

- genindex
- modindex
- search

2.1 Python installation

You can install the modules required to run APERO in three ways (eventually there will be a *setup.py* but not yet!

Currently supported options are:

- *install miniconda with the supplied environment* (recommended)
- *install miniconda and manually installing packages*
- *install anaconda with the supplied environment*
- *install anaconda and manually installing packages*
- *install via pip only*

Once python and the required modules are correctly installed you can install APERO - see *here*.

2.1.1 Installing miniconda (with supplied environment)

This is recommended for maximum compatibility

If you already use miniconda (with python 3) skip to step 3

Note: Make sure the miniconda you download/have is miniconda3

---

1. Download miniconda3, i.e. in bash and wget (or go to the anaconda website https:/repo.anaconda.com/miniconda/)
   i.e. the current latest version of Miniconda3 for Linux is this:

   ```
   wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
   ```

2. Install miniconda, i.e. with bash

   ```
   bash Miniconda3-latest-Linux-x86_64.sh
   ```

3. Find the APERO conda environment for you (you can either use the yml or the txt file) - note these files are system depenedent - do not install on non-supported systems

   ```
   cd apero-drs/setup/env/
   conda env create --file apero-env-{EXT}.yml
   ```

or

```
cd apero-drs/setup/env/
conda create --name apero-env --file apero-env-{EXT}.txt
bash apero-pip-{EXT}.txt
```

where {EXT} should be the latest version for your system / setup.

You should now have an environment called *apero-env*.

Before running or installing APERO you must be in this conda environment, i.e. type:

```
conda activate apero-env
```

2.1.2 Using miniconda (conda/pip install packages)

If you already use miniconda (with python 3) skip to step 3

Note: Make sure the miniconda you download/have is miniconda3

1. Download miniconda3, i.e. in bash and wget (or go to the anaconda website https:/repo.anaconda.com/miniconda/)
   i.e. the current latest version of Miniconda3 for Linux is this:

   ```
   wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
   ```

2. Install miniconda, i.e. with bash

   ```
   bash Miniconda3-latest-Linux-x86_64.sh
   ```

3. Create a new conda environment

   ```
   conda create --name apero-env python=3.7
   ```

4. Install the packages available via conda. You will need to check the currently supported packages in the *apero-drs/requirements_current.txt* file or for developers *apero-drs/requirements_developer.txt* file.

   ```
   conda install astropy=4.0
   conda install numpy=1.18.1
   ```

   Note: Some packages must be installed via pip

   ```
   pip install astroquery=0.3.10
   pip install barycorrpy=0.3.1
   ```

Note: If experiencing trouble with barycorrpy with the error: *Cannot remove entries from nonexistent* run the following:

```
pip install barycorrpy --ignore-installed
```

You should now have an environment called *apero-env*.

Before running or installing APERO you must be in this conda environment, i.e. type:

```
conda activate apero-env
```

2.1.3 Using anaconda (with supplied environment)

If you already use anaconda (with python 3) skip to step 3

Note: Make sure the anaconda you download/have is anaconda3

1. Download anaconda3, i.e. in bash and wget (or go to the anaconda website https:/repo.anaconda.com/archive/)
   i.e. the current latest version of Anaconda3 for Linux is this:

   ```
   wget https://repo.anaconda.com/archive/Anaconda3-2020.07-Linux-x86_64.sh
   ```

2. Install anaconda, i.e. with bash

   ```
   bash Anaconda3-2020.07-Linux-x86_64.sh
   ```

3. Find the APERO conda environment for you (you can either use the yml or the txt file) - note these files are system depenedent - do not install on non-supported systems

   ```
   cd apero-drs/setup/env/
   conda env create --file apero-env-{EXT}.yml
   ```

   or

   ```
   cd apero-drs/setup/env/
   conda create --name apero-env --file apero-env-{EXT}.txt
   bash apero-pip-{EXT}.txt
   ```

   where {EXT} should be the latest version for your system / setup.

You should now have an environment called *apero-env*.

Before running or installing APERO you must be in this conda environment, i.e. type:

```
conda activate apero-env
```

2.1.4 Using anaconda (conda/pip install packages)

If you already use anaconda (with python 3) skip to step 3

Note: Make sure the anaconda you download/have is anaconda3

1. Download anaconda, i.e. in bash and wget (or go to the anaconda website https:/repo.anaconda.com/archive/)
   i.e. the current latest version of Anaconda3 for Linux is this:

   ```
   wget https://repo.anaconda.com/archive/Anaconda3-2020.07-Linux-x86_64.sh
   ```

2. Install anaconda, i.e. with bash

   ```
   bash Anaconda3-2020.07-Linux-x86_64.sh
   ```

3. Create a new conda environment

   ```
   conda create --name apero-env python=3.7
   ```

4. Install the packages available via conda. You will need to check the currently supported packages in the *apero-drs/requirements_current.txt* file or for developers *apero-drs/requirements_developer.txt* file.

   ```
   conda install astropy=4.0
   conda install numpy=1.18.1
   ```

Note: Some packages must be installed via pip

```
pip install astroquery=0.3.10
pip install barycorrpy=0.3.1
```

Note: If experiencing trouble with barycorrpy with the error: *Cannot remove entries from nonexistent* run the following:

```
pip install barycorrpy --ignore-installed
```

You should now have an environment called *apero-env*.

Before running or installing APERO you must be in this conda environment, i.e. type:

```
conda activate apero-env
```

2.1.5 Manually using pip

1. install python 3.7 (install, via virtual environment etc)
2. pip install all modules. You will need to check the currently supported packages in the *apero-drs/requirements_current.txt* file or for developers *apero-drs/requirements_developer.txt* file.

```
pip install astroquery=0.3.10
pip install barycorrpy=0.3.1
```

Note: If experiencing trouble with barycorrpy with the error: *Cannot remove entries from nonexistent* run the following:

```
pip install barycorrpy --ignore-installed
```

2.2 Constants

DRS_ROOT
- This is the path where apero-drs was installed (via github)
- a suggested directory is `/home/user/bin/apero-drs`

DRS_UCONFIG
- The directory containing the users configurations files
- default is `/home/user/apero/`*`PROFILE`*

INSTRUMENT
- This is the instrument used at a specific telescope. Some settings are instrument specific.
- Currently supported instruments are:: SPIROU

PROFILE
- This is a short descriptive name given to a specific set of installation configurations
- Each profile contains setup files: *`PROFILE`*`.bash.setup file`, *`PROFILE`*`.sh.setup file`
- Each profile contains an instrument directory for each instrument. These contain user_config.ini and user_constant.ini files for said instrument.

2.3 Glossary

ds9

- An astronomical imaging and data visualization application
- see ds9.si.edu

pdflatex

- The pdf latex compiler
- see www.latex-project.org

2.4 Changelog

2.4.1 0.6.131 (2020-09-10)

- *Apero.recipes.spirou.pol_spirou.py* - hack from Issue #639 re: linear algebra error. [Neil Cook]
- Identical? [Neil Cook]
- *Apero.core.core.drs_startup.py* - format of splash update. [Neil Cook]
- Update date/version/docs/changelog. [Neil Cook]
- Update object query list. [Neil Cook]
- Issue #644 - deal with table = None in *generate_run_list* + add –test=True to codes which use processing *(obj_mk_tellu_db* an *dobj_fit_tellu_db)* [Neil Cook]
- *Apero.tools.module.setup.drs_processing.py* - deal with table being None (just to test if things work with this option) [Neil Cook]
- *Apero.recipes.spirou.obj_fit_tellu_db_spirou.py* - add break point and *TEST_RUN* = True for test. [Neil Cook]
- *Apero.io.drs_fits.py* - remove breakpoint. [Neil Cook]
- *Apero.io.drs_fits.py* - try to fix copying comments. [Neil Cook]
- *Apero.recipes.nirps_ha.cal_pp_master_nirps_ha.py* - move break point to test error. [Neil Cook]
- *Apero.io.drs_fits.py* - deal with header key not being str. [Neil Cook]
- *Apero.io.drs_fits.py* - deal with header key not being str. [Neil Cook]
- *Apero.recipes.nirps_ha.cal_wave_master_nirps_ha.py* – move breakpoint. [Neil Cook]
- *Apeor.data.nirps_ha.calib* - add *catalogue_UNe.csv* file. [Neil Cook]
- *Apero.science.extract.general.py* - remove breakpoint *apero.core.instruments.default.default_constants.py* - make *LEAKM_WSMOOTH* an int. [Neil Cook]
- *Apero.science.extract.general.py* - move breakpoint. [Neil Cook]
- *Apero.io.drs_image.py* - deal with only one image in *large_image_median* (return image without medianing) [Neil Cook]
- *Apero.recipes.nirps_ha.cal_shape_nirps_ha.py* – move break point. [Neil Cook]
- *Apero.recipes.nirps_ha.cal_shape_nirps_ha.py* – move break point. [Neil Cook]
- *Apero.recipes.nirps_ha.cal_shape_nirps_ha.py* – add break point. [Neil Cook]
- *Apero.recipes.spirou.obj_fit_tellu_spirou.py* + *science/telluric/fit_tellu.py* + *gen_tellu.py* + *mk_tellu.py* - fix problem with qc for tellu pre clean. [njcuk9999]
- Update version in readme for master/developer/working. [Neil Cook]
- Update date/version/changelog/update notes. [Neil Cook]
- Update date/version/changelog/update notes. [Neil Cook]
- Update date/version/changelog/update notes. [Neil Cook]

Index