GIANO-B reduction pipeline: GOFIO manual

Monica Rainer[1]

Affiliations:

[1]*INAF – Osservatorio Astrofisico di Arcetri*

Contact:

monica.rainer@inaf.it

Document version: 1.7

Date: 2020-07-13

If you use data reduced by GOFIO in your work, please reference the following paper:

Rainer M., et al., 2018, Proc.SPIE 10702, 1070266
Introducing GOFIO: a DRS for the GIANO-B near-infrared spectrograph
https://doi.org/10.1117/12.2312130

Change Record

Issue	Section	Page	Date	Observations
Issue 0	All		2017/11/20	Version 1.0
Issue 1	All		2017/12/11	Version 1.1
Issue 2	2.5,3.6,4	11,15,22	2018/03/02	Version 1.2
Issue 3	2.3,2.4,3.8,5	8,9,20,25,26	2018/05/04	Version 1.3
Issue 4	2.1,2.2,3.8,7	7,8,20,29,30	2018/06/27	Version 1.4
Issue 5	2,4.8,Appendix B	7,20,32,33	2019/04/03	Version 1.5
Issue 6	4.8,Appendix B	19,31	2019/05/23	Version 1.6
Issue 7	3,Appendix B	8,33	2020/07/13	Version 1.7

Table of Contents

Abbreviations and Acronyms

CCD	Charge Coupled Device
DRS	Data Reduction Software
FITS	Flexible Image Transport System
GIARPS	GIAno and haRPS
GUI	Graphical User Interface
HDU	Header Data Units
HWTM	Half-Width-at-Tenth-Maximum
MJD	Modified Julian Date
NaN	Not a Number
RON	Read-Out-Noise
SNR	Signal-to-Noise Ratio
TBC	To Be Confirmed
TBD	To Be Defined
TBW	To Be Written
TNG	Telescopio Nazionale Galileo
U-Ne	Uranium-Neon

Reference Documents

Ref.	Document title	Document ID
	GIANO_GUI_V1.pdf	
	GIANO_cookbook_for_observers.pdf	
	giarps_2D_reduction_v03_18jan2017.pdf	
	GIANO-B_Observer_Manual.pdf	

Table of Figures

1 Introduction

GOFIO is a DRS written for the GIANO-B infrared spectra. It is installed at the Telescopio Nazionale Galileo (TNG) and it is part of the online reduction pipeline running at the telescope: in this context, GOFIO interfaces with the ramp-processor database and reduces the data as soon as they are created. It can work also as a standalone offline DRS without a graphical interface, which allows to reduce the ramp-processed raw files available in the TNG archive.

GOFIO processes all the calibration files (darks, flat-fields and wavelength calibration lamps, both U-Ne and Fabry-Pérot) and the scientific images, observed both in the nodding and stare mode. The reduction process includes bad pixel and cosmic removal, flat-field and blaze correction, optimal extraction, wavelength calibration with U-Ne lamps, nodding or stare group processing. A logfile of the whole reduction process is created and stored in the reduction directory.

In the offline mode, the reduction process can be completely customized, but it is suggested to use only the command line options, unless having a great familiarity with the instrument and its settings.

This manual illustrates how to install, configure, and run GOFIO, and how the reduction process works. The outputs formats are briefly explained, and some issues sometimes found in the reduced spectra (independent from the reduction process itself) are shown.

2 Short overview (or GOFIO for dummies)

A short overview of the use of GOFIO is provided here, while the following sections will explain more in detail the installation, configuration and reduction process.

GOFIO may be used online (i.e. without any additional input) only at the TNG, because it needs to access specific databases. On any other occasion, GOFIO must be used in the offline mode, and by default it will reduce only the scientific spectra and not the calibrations. As such, it is important that the calibrations are explicitly reduced before the spectra. This is done because the wavelength calibrations (UNe lamps) are observed at the end of the night and reducing the data in the temporal order will result in using a previous wavelength calibration that may not be optimal for the night.

GOFIO works on a single whole night of observations. The raw data must be located in directories named yyyy-mm-dd. The root path to the raw and reduced directories must be hardcoded in the file config.py (see section 3.4). To use GOFIO, follow these steps:

1. reduce the calibration of the night, by calling GOFIO in this way:

   python gofioDRS.py yyyy-mm-dd --all_calib --only_calib

   If the calibration were observed out of order (the expected order being dark, flat-field, UNe lamps), then the following commands have to be issued instead:

   python gofioDRS.py yyyy-mm-dd --dark --only_calib

   python gofioDRS.py yyyy-mm-dd --flat --only_calib

   python gofioDRS.py yyyy-mm-dd --une --only_calib

2. if any calibration fail the quality check, it will not be used in the reduction. Be sure that at least a calibration per type (dark, flat and UNe lamp) has been successfully reduced before working with the scientific images.

   Be aware that using an UNe lamp taken in a different night than the spectra may result in an incorrect wavelength calibration, while using flat-field images of a different night may result in an incorrect definition of the echelle orders.

   The former problem may be solved using the telluric lines as wavelength calibrators and the latter by checking the straightened images (stored in the STR subfolder of the reduced night directory) and passing directly the vertical shift to GOFIO following the procedure described in step 5 of section 4.6.

3. once the desired calibrations have been reduced and stored in the calibration database, the spectra may be reduced using the command line:

   python gofioDRS.py yyyy-mm-dd

4. for any other option (no blaze removal, no flat-field removal, normalized merged output spectra, group reduction and so on), please read the manual and use the GOFIO help:

   python gofioDRS.py -h

GOFIO will give continuous information on the reduction on the terminal and it will save a log of the reduction (drs.log) in the reduced directory.

3 Requirements, installation and configuration

GOFIO can be downloaded from the webpage https://atreides.tng.iac.es/monica.rainer/gofio.

Be sure to download the latest stable version (master). There are two versions: the Python2.7 version in the gofio directory, and the Python3 version in the gofio3 directory.

Accordingly to the version downloaded and the computer default settings, all the commands shown in this manual may have to be changed from "*python <command>*" to "*python2 <command>*" or "*python3 <command>*".

3.1 Requirements

GOFIO is written almost completely in Python, with the exception of a single fortran77 subroutine.

GOFIO requires the following python packages (the version must be at least the indicated one):

- NumPy v1.12
- SciPy v0.19
- AstroPy v1.3
- ccdproc v1.2 (a sub-package of AstroPy)
- watchdog v0.8.2
- docopt v0.6.1
- barycorrpy v0.2

The fortran77 program (straight_giano_2D fortran) was written by E. Oliva and it is used to straighten the echelle orders. A fortran77 compiler is needed (either f77 or gfortran are suitable) and the library libcfitsio must be installed.

3.2 Software modules and structure

To run the GOFIO pipeline, all the following files and directories are needed (optional files are shown between brackets):

gofio/ OR gofio3/

__init__.py: to define the module directory;

gofioDRS.py: the main program;

gofio.cfg: user configuration file (optional);

__version__.py: where the version number of GOFIO is stored

drslib/

__init__.py: to define the module directory;

(clean_db.py: auxiliary file to remove unused calibration files from the calibration directory);

config.py: where all of the configuration parameters are written;

darkframes.py: subroutines for the processing of the dark frames;

db.py: setting and using the connection with the calibration database, the ramp processor database and the night's reduction database;

flatframes.py; subroutines for the processing of the flat-field frames;

gofio.py: subroutines to initialize and run GOFIO;

(gofio2ascii.py: auxiliary program to convert the output files from FITS to ASCII);

logger.py: the GOFIO logger;

nodding.py: subroutines for the processing of the nodding frames;

rawfiles.py: subroutine to identify the observed frames and decide their reduction process;

stare.py: subroutines for the processing of the stare frames;

varie.py: various subroutines (bad pixel removals, optimal extraction, wavelength calibrations, and so on);

wlframes.py: subroutines for the processing of the wavelength calibration lamps (U-Ne and FP);

resources/

badpix_mask.fits: positions of the bad pixels on the detector;

(GIANOB_MASKC.fits: not necessary, GOFIO will rewrite it every time);

straight_giano_2D_v1_1.f : fortran77 program, it will have to be compiled with the following syntax *gfortran/f77 -o straight_giano_2D straight_giano2D_v1_1.f -lcfitsio*;

Une_lines_GIANO_selected.txt: selection of very bright U-Ne lines whose wavelength and pixel position are well known;

Une_observed_lines_GIANOB_18nov2016.txt: all the U-Ne lines that have been identified.

3.3 Installation

A setup tool will be provided in due time. In the meanwhile, it is possible to manually set up GOFIO by following these steps:

- preserve the directory structure shown in the above sub-section;
- compile the fortran subroutine straight_giano_2D_v1_1.f in the resources directory using a fortran77 compiler such as gfortran. The syntax is the following:

  gfortran -o straight_giano_2D straight_giano_2D_v1_1.f -lcfitsio

  The CFITSIO library may be found at: https://heasarc.gsfc.nasa.gov/fitsio/fitsio.html

  If the CFITSIO library is installed in a directory that is not present in the default library path, it must be specified during the compiling:

  gfortran -o straight_giano_2D straight_giano2D_v1_1.f -L/PATH_libcfitsio -lcfitsio
- install all the required python packages.

The newest version of the python packages may need python3, so they must be used with the gofio3 version. If only python2 is available, please install the older version using the command: pip install ‘package_name<version_number’ (*e.g.* pip install ‘ccdproc<1.3’), where the version number may be found in subsection 3.1.

If you have problems installing the python packages because of directory permits or operating systems python installation, you could install GOFIO in a virtual environment.

3.4 Configuring and launching GOFIO

All the configuration parameters may be found in the configuration file drslib/config.py (see appendix). This file should never be modified, except for the following two parameters:

- CONFIG['BASE_RAW']: base raw directory, where GOFIO expects to find the raw files in yyyy-mm-dd directories;
- CONFIG['BASE_RED_DIR']: base reduced directory, where GOFIO will write the reduced files in yyyy-mm-dd directories. If it not specified, GOFIO will create a reduced directory in

GOFIO will run in online mode only at the TNG, where it expects to connect with the database of the ramp processor, whose path is defined by the parameter CONFIG['BASE_RAMP'].

Outside from the TNG, GOFIO must be run only in the offline mode. In this case, it is possible to define a user configuration file gofio.cfg. The parameter OFFLINE:::True must be set, otherwise the rest of the setup will not be used. All the configuration parameters found in drslib/config.py may be overridden by re-defining them in the user configuration file. The syntax to follow is *configuration_key:::value*.

Alternatively, it is possible to run GOFIO in offline mode with minimal changes to the default configuration by command line. Run "*python gofioDRS.py -h*" for help on the different available options.

```
~/Documents/gofio/gofio$ python gofioDRS.py -h
GOFIO DRS
Written by Monica Rainer

Usage:
    gofioDRS.py -h | --help
    gofioDRS.py [(-g <config> | --cfg=<config>)]
    gofioDRS.py <date> [(-g <config> | --cfg=<config>)]
    gofioDRS.py <date> <calib_date> [(-g <config> | --cfg=<config>)]
    gofioDRS.py <date> [--dark --flat --une --fp --all_calib --only_calib --use_
flat=<flag> --s1d=<s1d> --group]
    gofioDRS.py <date> <calib_date> [--dark --flat --une --fp --all_calib --only
_calib --use_flat=<flag> --s1d=<s1d> --group]

Options:
    -h,--help            : show this screen
    --cfg=<config>       : path of the configuration file (optional)
                           'filepath/filename'
    -g <config>          : path of the configuration file (optional)
                           'filepath/filename'
    date                 : date to be reduced (always first input)
    calib_date           : calibration date to be used (always second input)
    --dark               : darks are reduced
    --flat               : flats are reduced
    --une                : U-Ne lamps are reduced
    --fp                 : FP lamps are reduced
    --all_calib          : all the calibrations are reduced
    --only_calib         : only calibrations are reduced, no science images
                           (the calibrations to be reduced must be defined
                            using the above parameters)
    --use_flat=<flag>    : flat-field is removed,
                           options are global/order/nor/none [Default: order]
    --s1d=<s1d>          : s1d outputs are created, either normalized or not
                           options are yes/no/norm [Default: yes]
    --group              : group reduction of all the images of the same
                           observing block
```

Figure 1: Screenshot of the GOFIO help.

To summarize, GOFIO may be run mainly in four different configurations:

- *python gofioDRS.py* → online mode, only at the TNG;
- *python gofioDRS.py --cfg=filepath* → offline mode with an user configuration file. The parameter OFFLINE:::True must be set. If the user configuration file is named gofio.cfg and it is located in the same directory as gofioDRS.py, then it will be read automatically even without the option "--cfg=filepath";

- *python gofioDRS.py yyyy-mm-dd* → offline mode, only the scientific images of the night yyyy-mm-dd will be reduced using the calibration database;
- *python gofioDRS.py yyyy-mm-dd …* → offline mode with additional parameters as defined in the GOFIO help.

3.5 First use of GOFIO

The first time it is run, GOFIO will create these additional directories if they do not exists yet:

gofio/calibrations/

gofio/calibrations/calfiles/

gofio/calibrations/database/

gofio/webuidatabases/

CONFIG['BASE_RED_DIR']/yyyy-mm-dd/(offline/)

CONFIG['BASE_RED_DIR']/yyyy-mm-dd/(offline/)CALIB/

CONFIG['BASE_RED_DIR']/yyyy-mm-dd/(offline/)STR/

where offline/ is added only if GOFIO is launched in offline mode (either by using the user configuration file gofio.cfg or by command line).

Additionally, it is important to notice that the calibration database will be empty when using GOFIO for the first time. As such, it is important to populate the database with at least one calibration per type before processing the scientific spectra, so that GOFIO will always find a default image to use in the reduction. To do so, two strategies can be followed:

- if the night calibrations have been taken in the correct default order of darks, flat-fields and U-Ne lamps, it is sufficient to give the command:

  python gofioDRS.py yyyy-mm-dd –all_calib –only_calib

  Every time an observing night is reduced offline it is recommended to run the above command in order to reduce all the night calibrations, and then re-run GOFIO with *python gofioDRS.py yyyy-mm-dd* in order to reduce all the scientific spectra;
- if the calibrations have not been observed in the correct order, it is necessary to reduce them separately:

  python gofioDRS.py yyyy-mm-dd --dark --only_calib

  python gofioDRS.py yyyy-mm-dd --flat --only_calib

  python gofioDRS.py yyyy-mm-dd --une --only_calib

At this point, if the reduction has been successful, the calibration database has been populated. Otherwise, GOFIO will issue some warnings and other calibrations will have to be used.

4 Reduction process

The reduction process will be explained in detail in the following sections.

The sequence of actions is this: GOFIO will create or connect to the relevant databases, then it will search for the ramp-processed raw files in the target directory and it will reduce them in order by group of images and by type (dark, flat-field, U-Ne lamps, Fabry-Pérot lamps, scientific images).

The outputs are saved in the reduced directory, and the calibrations are also saved in a dedicated directory and inserted in a calibration database, that is queried every time a calibration is needed.

A dedicated database monitors the reduction process, and it is used by the online version of GOFIO to inject the data into the TNG archive and to consult in case the reduction is interrupted and then re-started in order to not reduce again the same files.

4.1 Initialization (drslib/gofio.py)

The default configuration file config.py, the user configuration file gofio.cfg and/or the command line configuration input are read and processed.

GOFIO is now running with the desired configuration.

4.2 Logfile (drslib/logger.py)

A logfile drs.log is created in the reduction directory. Every message and warning issued by GOFIO is recorded in this file. If the data are re-reduced in the same reduction directory, the information on the new reduction will be appended to the existing logfile, without erasing any previous information.

4.3 Databases (drslib/db.py)

GOFIO connects to/creates/manages three databases:

1. the ramp database, that is accessed only by the online version at the TNG in order to check that the raw FITS files are completely written;
2. the calibration database, where GOFIO writes the paths to the reduced calibrations for each night, in order to use them in the offline version (or when needed) without having to reduce them again;
3. the reduction database, that can be named either db_yyyy-mm-dd_online.db or db_yyy-mm-dd_offline.db. This database monitors the reduction process and it is used to inject the files in the archive. It contains three tables:
    a) spec2dfiles: where the raw SCIENCE files are listed in group or pairs when they are ready to be reduced. It contains two columns: a comma separated list of the raw files paths and the unique time-stamp of the start of the reduction;
    b) spec1dfiles: the path of each of the reduced SCIENCE file is stored in this table along with the times-tamp of the associated raw files, an unique identifier, the object name, the signal-to-noise ratio in the Y, J, H, and K band, the slit position, and some other additional information;

c) calib: only for calibration files, it contains a comma separated list of the paths of a group of raw calibration files, their calibration type (DARK, FLAT, WCAL_UNE, WCAL_FP), a comma separated list of the quality check flag (OK or FAILED), and the unique time-stamp of the start of their reduction process.

4.4 Frame control (drslib/rawfiles.py)

Each ramped raw files is read by rawfiles.py. This subroutine check their type (*i.e.* the value of the keyword OBS-TYPE, that may be DARK, FLAT, WCAL_UNE, WCAL_FP, or SCIENCE) and their belonging to a group.

If the group is complete, or another image of a different group arrives, the group is reduced according to their type. In case of SCIENCE images, the reduction depends also on their observing mode (nodding or stare).

4.5 Dark frames (drslib/darkframes.py)

Once a complete group of dark frames has been observed, GOFIO reduced them. With an observing block of n dark frames, GOFIO expects to read 7*n files, because the darks are ramped in such a way to create raw files with different exposure times (10, 30, 60, 100, 200, 300, and 600 s). If not all of them are present, GOFIO will simply reduce the existing ones.

Each dark frame undergoes a quality check: if its exposure time is different than the default ones or the average signal in at least one of the quadrants of the image is larger than the expected threshold, the frame is discarded (see Fig. 2 for the detector's quadrants).

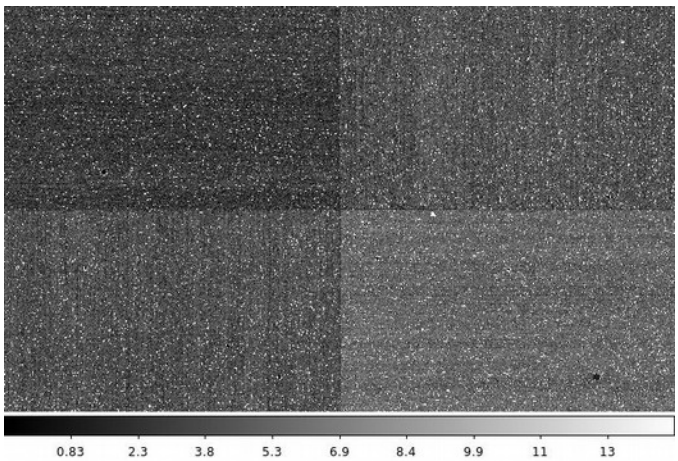

Figure 2: 100 seconds masterdark, showing the difference between the quadrants of the detector.

If enough frames of a particular exposure time have passed the quality check, they are averaged to create a masterdark. The masterdark is then inserted in the calibration database. If not, the masterdark for that exposure time will be taken from the calibration database according to the nearest date and a warning message will be given.

All the reduced dark frames are saved also in the CALIB directory nested inside the night reduction directory.

4.6 Flat-field frames (drslib/flatframes.py)

As soon as a group of flat-fields has been observed, they are reduced together.

Each flat-field undergoes a quality check by computing the mean signal over all the image and comparing it to a threshold value: if the value is lower than the threshold, the frame is discarded.

If no flat-field passes the quality check, the masterflat is taken from the calibration database and a warning message is given, otherwise the following reduction steps are performed:

1. the masterdark with the appropriate exposure time is subtracted;
2. the bad pixels are removed using the bad pixel mask: the frame is filtered using a rectangular filter 41-pixel wide and 1-pixel high and then the bad pixel mask is used to define the bad pixel positions and substitute their values with the filtered ones (see Fig. 3);

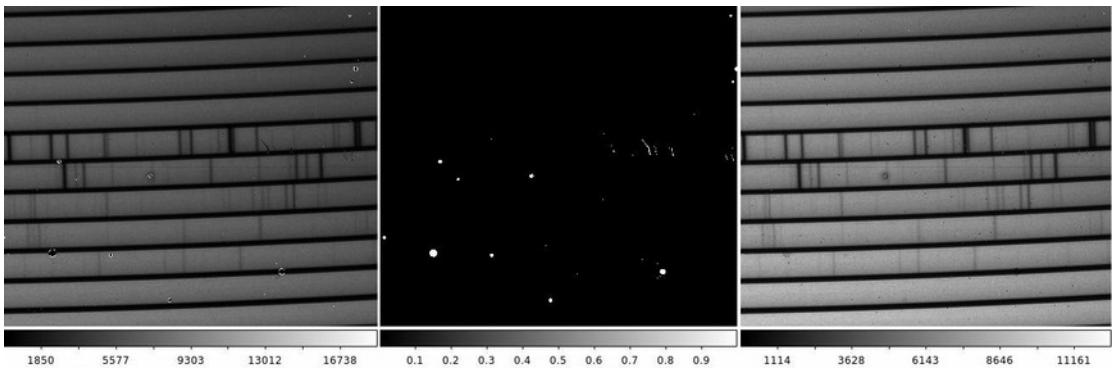

Figure 3: Flat-field with bad pixels (left), bad pixel mask (center), and flat-field with bad pixels removed (right)

3. the flat-fields are combined to create the masterflat:
   a) the flat-fields are masked by sigma-clipping using the median and the standard deviation modified by the read-out-noise (RON) and gain;
   b) the sigma-clipped flat-fields are averaged to create the masterflat;
   c) RON and gain of the resulting masterflat are adjusted accordingly;
4. the vertical position of the orders on the detector is computed by finding the bottom pixel of the flat signal in the lowest order (order 32) and the leftmost column of the detector. The value is then stored in the calibration database to be used in the following step;
5. the echelle orders of the masterflat are straightened using a fortran77 procedure courtesy of E. Oliva, which takes into account the slit-tilt and interpolate the signal. The curvature of each order can be represented by a parabolic function:

$$Y(X,N) - Y_0(N) = C(N) \cdot [(X - X_0(N))/2048]^2$$

   where the parameters are:
   a) N is the echelle number (from 32 to 81);
   b) X is the horizontal pixel coordinate (from 1 to 2048);
   c) Y(X,N) is the vertical pixel coordinate of the bottom of the trace at the given X coordinate;
   d) $Y_0(N)$ is the vertical pixel coordinate of the minimum of the order;

e) $X_0(N)$ is the horizontal position of the minimum of the order, and can be approximated as $X_0(N) = 833.2 - 24.51 \cdot N + 230.3 \cdot \mathrm{sqrt}(N-25)$;

f) C(N) is the curvature parameter, defined as $C(N) = 120.5 + 0.765 \cdot N$

The geometry of the orders is an intrinsic property of the GIANO-B spectrometer and it is always the same, but a shift of the spectral image (see Fig. 4) may happen following major maintenance work of the cryogenic spectrograph (e.g. warm-up cycles, or manual movements of the slit wheel). As such, the straightening procedure accepts as input parameter a vertical shift in pixels (DY).

This parameter may be given directly from the configuration files (config.py or gofio.cfg), otherwise it will be taken from the calibration database (see previous step). It is important to notice that DY is not an absolute pixel value, but the difference between the orders' position and the default position used by the straighten program, which is stored in the configuration value CONFIG['SHIFT_Y'];

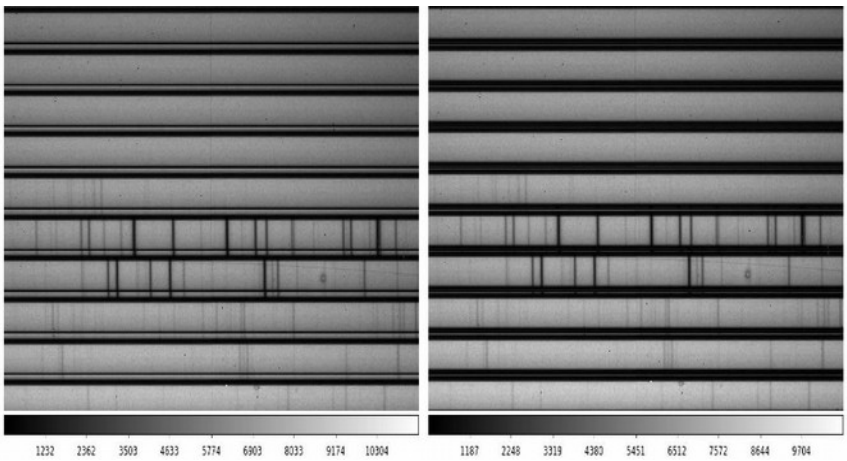

Figure 4: The narrow black lines in the left image define the order regions and they should fall in between orders (right image). A vertical shift of the spectral image may happen from time to time.

6. the straightened masterflat is normalized by fitting each row with a 5th order degree polynomial and then dividing the row by the polynomial (see Fig. 5);

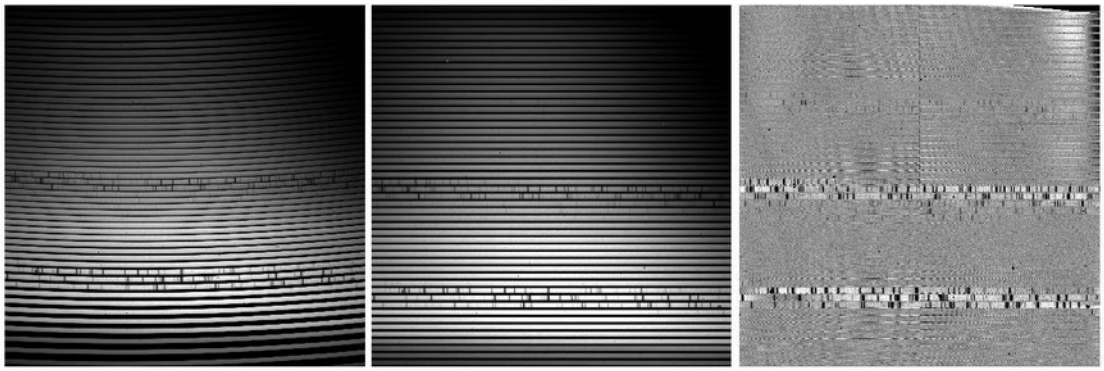

Figure 5: masterflat (curved orders, left), straightened masterflat (center), and normalized masterflat (blaze removed, right)

7. the straightened masterflat is used to create the extraction mask with the position and the width of the echelle orders;

8. masterflat, straightened masterflat and normalized masterflat are inserted in the calibration database.

All the reduced flat-field frames are saved also in the CALIB directory nested inside the night reduction directory.

4.7 Wavelength calibration lamps (drslib/wlframes.py)

GIANO-B has two different calibration lamps: an U-Ne lamp and a Fabry-Pérot.

At the moment, the stability of the Fabry-Pérot is still being studied, and so only the U-Ne lamp is used for the wavelength calibration.

4.7.1 U-Ne lamp

It is important to remember that the U-Ne frames have to be observed several hours before the observations or immediately after, because there is a problem of persistence of the lamp's signal on the detector.

As such, the online version of GOFIO will calibrate the scientific spectra with a U-Ne exposure that may not be the best choice, because of the time gap between the U-Ne observation and the science observations.

As soon as the observing night is finished, it is advisable to observe a last U-Ne, wait for the reduction to end and then re-launch GOFIO in offline mode. In this way, GOFIO will re-reduce the whole night using the last observed U-Ne lamp as a wavelength calibrator.

The reduction of the U-Ne lamps follows these steps:

1. each frame undergoes a quality check TBD;
2. the appropriate masterdark is subtracted from each U-Ne frame;
3. the bad pixels are removed using the bad pixel mask (see point 2 of flat-field reduction);
4. if there is more then one U-Ne frame, they are combined after sigma-clipping them (see point 3 of flat-field reduction), creating the masterlamp;
5. the masterlamp is straightened (see point 5 of flat-field reduction);
6. 20 central pixels of each order are extracted and a default wavelength calibration is performed (see Fig. 6, and read below for details);

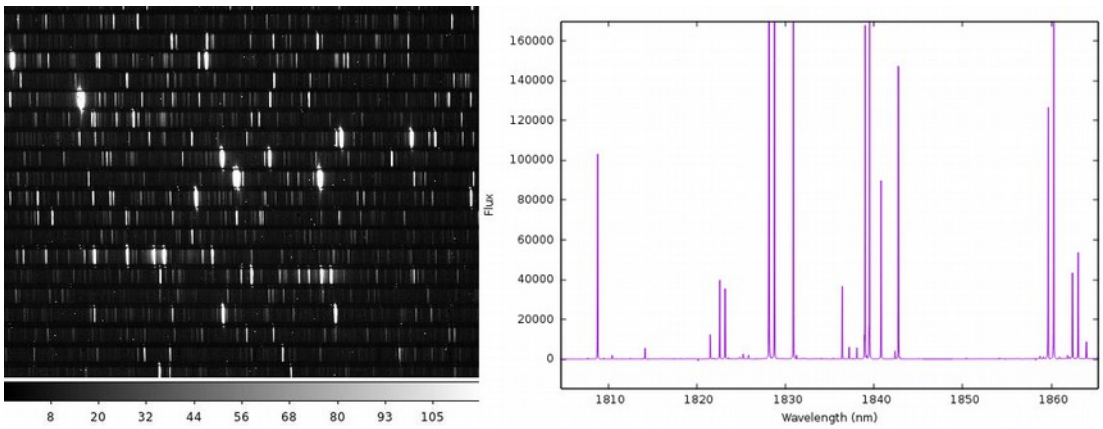

Figure 6: Straightened masterlamp (left) and extracted and calibrated masterlamp (right)

7. the extracted and calibrated U-Ne masterlamp is inserted in the calibration database.

The wavelength calibration is done using an atlas of U-Ne lines courtesy of E. Oliva. We selected as a starting point 743 prominent and isolated lines for which both the position on the detector and the wavelength are well known.

For each echelle order, the lines are found using the pixel position, fitted with a gaussian and discarded if the fit fails or if the parameters of the gaussian are very far from the expected central position and sigma. The remaining lines are used to perform a preliminary calibration using a modified 3rd order degree polynomial (courtesy of E. Oliva):

$$\lambda = \lambda_0 + k_1 \bullet (x-x_c) + k_2 \bullet (x-x_c)^2 + k_3 \bullet (x-x_c)^3$$

where k_1, k_2 and k_3 are computed for each echelle order using the following equations:

$$k_1 = -0.849(1/\text{order} - 1/2150)$$

$$k_2 = -3.560 \bullet 10^{-5} / \text{order}$$

$$k_3 = 1.780 \bullet 10^{-9} / \text{order}$$

with the order numbers going from 32 to 81.

This polynomial leaves us with only two free parameters (λ_0 and x_c), which is very useful considering that some echelle orders have very few U-Ne lines available.

Once this first calibration is done, GOFIO searches for all the other lines in the atlas and repeat the gaussian fitting and quality check, and then the calibration. The polynomial parameters and the RMS of the fit are written in the header of the extracted masterlamp FITS file for each echelle order.

If the calibration fails, a warning message is given: the user may either observe another lamp or fall back to the calibration database.

As a sanity check, at this moment GOFIO calibrates in wavelength also using a standard 3rd degree order polynomial ($\lambda = c_0 + c_1 \bullet x + c_2 \bullet x^2 + c_3 \bullet x^3$), and stores also this calibration coefficients and RMS in the header of the extracted masterlamp.

It is important to remember that for the calibration process the first pixel is pixel number 1 and not pixel number 0.

All the reduced U-Ne frames are saved also in the CALIB directory nested inside the night reduction directory.

4.7.2 Fabry-Pérot

The stability of the Fabry-Pérot is still being studied, as such these frames are reduced but not used for the wavelength calibration. The U-Ne calibration is instead written in the FP masterlamp header.

All the reduced Fabry-Pérot frames are saved also in the CALIB directory nested inside the night reduction directory.

4.8 Nodding pairs and groups (drslib/nodding.py)

The main GIANO-B observing mode for point sources is the nodding mode. The object is observed first in nodding position A, then B (or B and then A), and the reduction starts when the nodding is complete. In Fig. 7 the raw images of a nodding are shown.

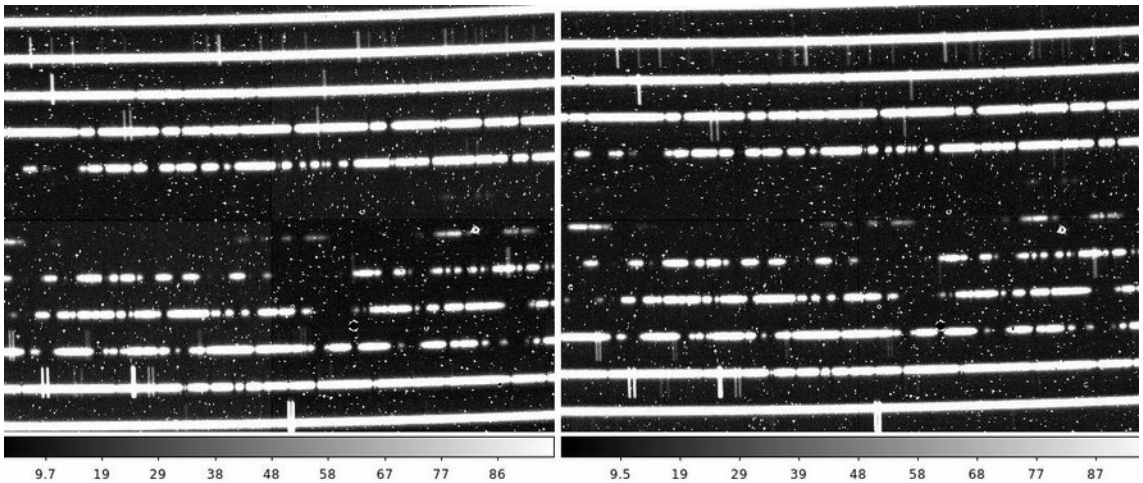

Figure 7: Raw images of star observed in nodding A (left) and B (right).

The nodding undergoes a check to be sure that the exposure times of the two frames are the same, otherwise they will not be reduced.

Once the images pass the quality check, the reduction follows these steps:

1. the bad pixels are removed (see point 2 of the flat-field reduction);
2. the image A-B is created, which takes care of the sky and dark subtraction (see Fig. 8). From this point on all the reduction is performed on this image, while the separate A and B images are used only for their headers;

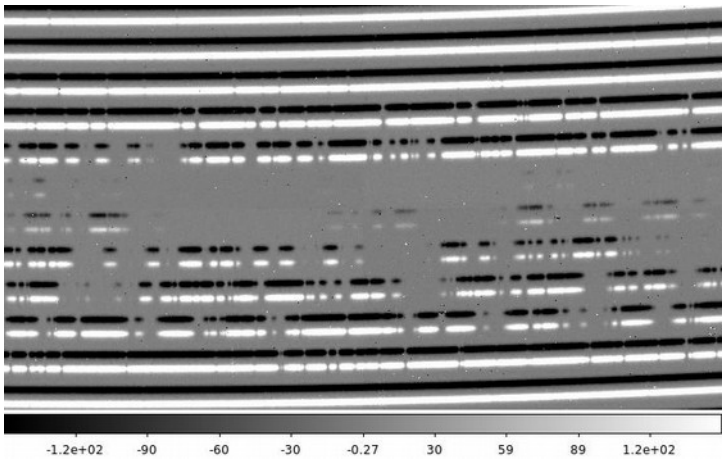

Figure 8: Nodding A-B: the B trace has negative values.

3. the image is straightened (see point 5 of flat-field reduction). The straightened images are saved in the STR directory nested in the night reduction directory;
4. the image is divided by the straightened masterflat. By default, the division is performed on the 2D images order by order: each order region of the masterflat is first divided by its mean value, then the same order region of the nodding is divided by this normalized masterflat. The values of RON and gain are updated accordingly. In the offline version of GOFIO it is possible to choose different procedures: either divide by the masterflat normalized by its global mean value, or divide by the normalized

masterflat (and skip the blaze correction, see point 6 of the flat-field reduction), or not divide by the masterflat at all;

5. extraction of the signal A from A-B and the signal B from -(A-B) with an optimal extraction algorithm (Horne, K., 1986, PASP 98, 60, *An optimal extraction algorithm for CCD spectroscopy*). The extraction is done order by order and follows these steps:

   a) a mean order profile is created by averaging the order columns along the whole order length and setting all the negative, infinite or NaN values to zero;

   b) the order profile is fitted by a gaussian. If the x_0 position of the gaussian is very different from the expected position of the object on the slit, the expected position is used. If the half-width-at-tenth-maximum (HWTM) is smaller than 3 pixels (probably a cosmic is driving the fit) or larger than the configuration value (risk of overlapping between A and B traces), then the configuration value is used;

   c) the x_0 and the HWTM of the gaussian profile are used to define the extraction window and to perform a standard extraction. The results are the standard flux and its variance. A mean of the x_0 values and of the borders of the extraction window on all the orders is saved in the FITS header of the reduced files;

   d) a spatial profile is built by dividing the data for the standard flux. The profile is then corrected by setting all the negative, infinite and NaN values to zero, and normalized;

   e) the spatial profile is optimized by fitting it row by row with a 2nd degree order polynomial, weighting the signal with inverse of the square root of the variance. All the points more than 4-σ from the fit are rejected and substituted with the fitted values (see Fig. 9);

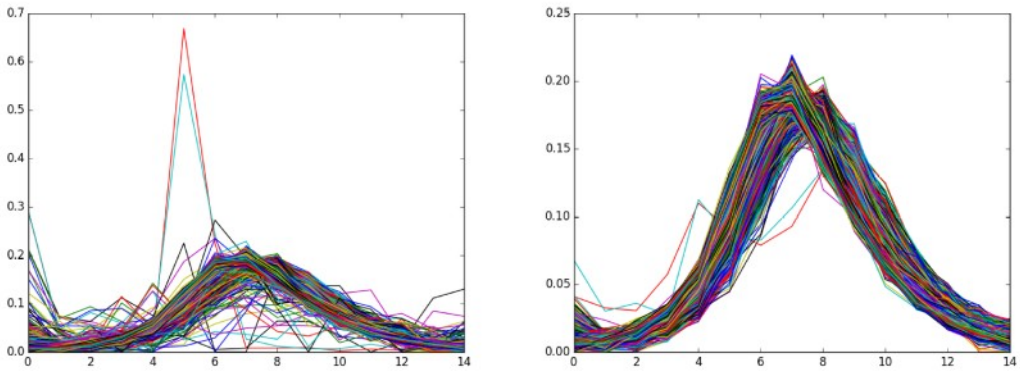

Figure 9: Sample extraction profile for order 32 before (left) and after (right) the optimization.

   f) the optimized profile is corrected for negative, infinite and NaN values, and normalized, the variance is updated;

   g) a first optimal extraction is performed with the optimized profile: the results are the optimized flux and its variance;

   h) a 2D model of data is created by multiplying the optimized profile by the optimized flux;

   i) the cosmic rays and other outliers are removed iteratively:

- the outliers are found by comparing the difference between the data and the model with the square root of the variance;
- only the strongest outlier is removed each time, and only if the above absolute value is greater than 5 . The outlier is removed by substituting it with the median of a small interval around it, the same is done for the extraction profile in correspondence to the outlier;
- the process is repeated until no more outliers are found or the maximum number of outliers per order (50) is reached;

j) the final optimal extraction is performed on the clean data;

6. the signal-to-noise ratio is computed for each pixel of the extracted spectrum. If the masterflat has been used, it will be extracted with the same optimized profile in order to correct the signal-to-noise ratio. The formulas used are the following:

$$SNR = \frac{1}{\sqrt{\epsilon_{sp}^2 + \epsilon_{flat}^2}}$$

where:

$$\epsilon_{sp} = \frac{\sqrt{(RON_{eff}^2 + (gain_{eff} \cdot flux))}}{gain_{eff} \cdot flux}$$

$$\varepsilon_{flat} = \frac{\sqrt{(fRON_{eff}^2 + (fgain_{eff} \bullet fflux))}}{(fgain_{eff} \bullet fflux)}$$

with:

RON_{eff}, $gain_{eff}$, = effective RON and gain of the spectrum;

flux = flux of the extracted spectrum;

$fRON_{eff}$, $fgain_{eff}$ = effective RON and gain of the flat-field;

fflux = flux of the extracted flat-field;

7. the same extraction profiles are used also to extract signal from the calibration masterlamp, then the extracted masterlamp is calibrated and the wavelength solution applied to the spectra. If the calibration fails, the default wavelength solution found at point 6 of the U-Ne lamp reduction will be used;
8. the wavelength calibrated A and B nodding are averaged to create the AB spectrum, adjusting RON, gain, slit position, airmass, and exposure time;
9. the barycentric correction is computed for each spectrum, both in radial velocity and in time. The python module barycorrpy is used to compute the velocity correction and the barycentric Julian Date (taking into account also the proper motions of the target), while AstroPy is used to compute the heliocentric Julian Date. The first time it is run, barycorrpy will download the JPL ephemeris.

   The result are stored in three different keywords: HIERARCH TNG DRS BERV (velocity correction in km/s), HIERARCH TNG DRS HJD (mid-exposure heliocentric Julian Date, UTC), and HIERARCH TNG DRS BJD (mid-exposure barycentric Julian Date, TDB);

10. all the 3 spectra (A, B, and AB) are saved in two different formats:
    a) *_ms1d.fits: binary table with 50 rows (one row for each order) and 4 columns (wavelength, flux, SNR, order number from 32 to 81). All the relevant information are in the header of the primary HDU, while the table itself is the secondary HDU. The spectra have constant step in pixel, they are all 2048 pixel long. The barycentric correction is stored in the header, but it is not applied;
    b) *_s1d.fits: monodimensional fits image with a constant step in wavelength of 0.001 nm, with the barycentric correction applied;
11. in the case of the AB spectrum, the exposure times, the Modified Julian Date (MJD), and names of the A and B images are stored in the header.

In the offline mode, it is possible to set GOFIO to reduce a whole group of noddings (belonging to the same observing block. Using the flag “--group”, GOFIO will reduce first each nodding as it arrives, and then the whole group in the following way:

1. all the raw A noddings are averaged, adjusting RON, gain, and airmass. All the exposure times and MJD of the combined images are saved in the header;
2. all the raw B noddings are averaged, adjusting RON, gain, and airmass. All the exposure times and MJD of the combined images are saved in the header;
3. from here on, the reduction follows the standard steps of nodding reduction, with the average A as the nodding A and the average B as the nodding B;
4. the final output will have a 'grp.fits' suffix.

4.9 Stare groups (drslib/stare.py)

As soon as a sequence of stare images and associated sky images (if any) has been observed, GOFIO reduces the whole group. The reduction follows these steps:

1. the exposure times of the target images are checked: if the images have different exposure times, the ones with the most common exposure times are kept, the others are discarded;
2. the exposure times of the sky images are checked: if different from the target images, they are discarded;
3. an average target image is created, the values of RON, gain, and air-mass are adjusted accordingly;
4. an average sky image is created, adjusting the values of RON and gain;
5. the average sky image is subtracted from the average target image. If there is no available sky image, a DARK is used instead. If there is no DARK available, the reduction will continue anyways. A warning will be given for each of these situations. The values of RON and gain are adjusted accordingly;
6. the names and MJDs of every target and sky image used are written in the header of the sky-subtracted image;
7. the sky-subtracted image is straightened (see point 5 of flat-field reduction);
8. the image is divided by the masterflat (see point 4 of nodding reduction);

9. the spectrum is extracted (see point 5 of nodding reduction). The only difference with the nodding extraction is on the limits of the extraction profile, which in this case may span the whole slit width (see Fig. 10);
10. same as points 6-10 of the nodding reduction process.

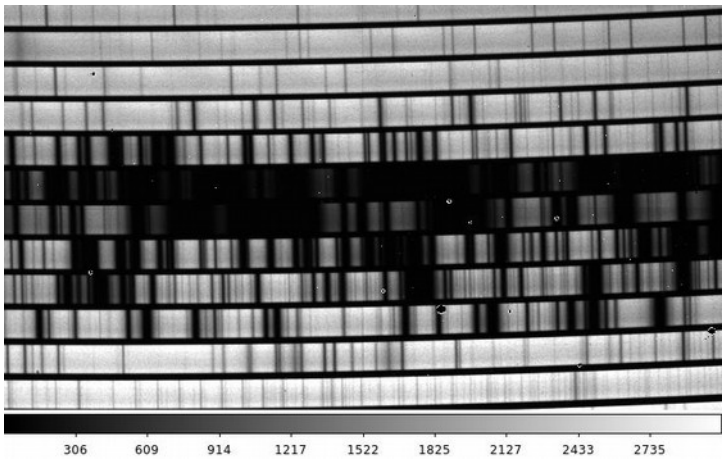

Figure 10: Raw stare image of Mars: the signal covers the whole slit width.

5 Output formats

The reduced spectra are available in two different formats. In both cases the spectra are fully reduced, *i.e.* the sequence shown in the nodding and stare sections was followed.

The formats are:

1. *_ms1d.fits: the multi-spectral 1D output, where the echelle orders are kept separated and the step in pixel is preserved. The output is a FITS bintable with 50 rows (one for each echelle orders).

   There are 4 columns: the number of the echelle order (from 32 to 81), the wavelength (nm), the flux, and the SNR. All the keywords are stored in the header of the primary HDU. See Fig. 11 for the structure of this output;

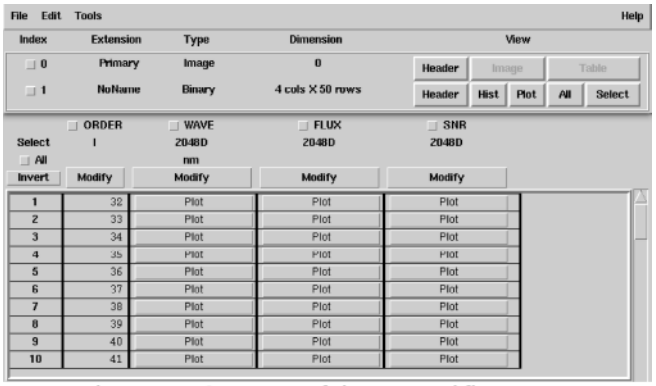

*Figure 11: Structure of the *_ms1d.fits output.*

2. *_s1d.fits: the single spectrum 1D output, with the echelle orders merged and the spectrum rebinned to a constant step in wavelength (0.001 nm).

   The output is a FITS image, with all the relevant keywords in the header of the primary HDU (see Fig. 12). The wavelength information can be retrieved from the CRVAL1 (wavelength of the first pixel) and the CRDELT1 (step in wavelength) keywords.

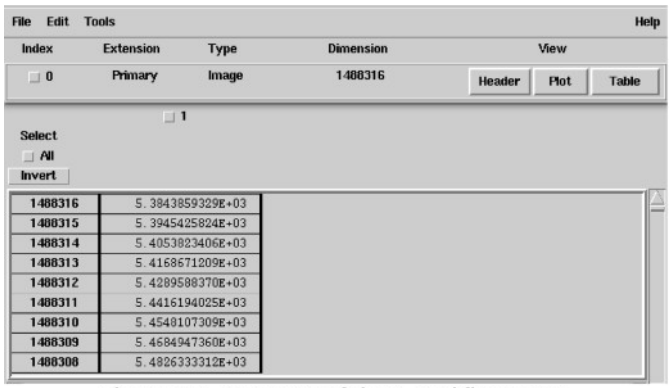

*Figure 12: Structure of the *_s1d.fits output.*

The output files may be converted from FITS to ASCII using the drslib/gofio2ascii.py program. The program needs a text list of the FITS files to convert as an input and it will query if the *_ms1d.fits files are to splitted in the 50 orders (and thus 50 files will be created from each spectrum).

The *_s1d.fits files are converted in two-column ASCII files (wavelength and flux), while the *_ms1d.fits files are converted in three-column ASCII files (wavelength, flux and SNR).

6 Possible spectral issues and how to solve them

It is important to remember than any change in the configuration reflects on the whole of the reduction process. As such, when changing a parameter, all the calibration files have to be reduced again before working the the scientific images.

When working with time-series it is crucial to use the same configuration for all the observed spectra.

The reduced spectra may show some peculiar behaviors in particular circumstances:

- the signal in the blue part of the blue orders is very low, and it is particularly notable when dividing by the flat-field (and when the SNR is low), because the blaze function as defined by the flat-field will not be optimal in these regions.

  The flat-field division will then cause a sharp decrease/increase of the signal in these regions: the AB spectra are less affected, because the effect is often averaged out on the A and B nodding.

  The compensate this effect, it is possible to use the normalized flat (python gofioDRS.py yyyy-mm-dd –use_flat=nor) and then remove the blaze function in some other way.

  Figure 13 shows this effect in one of the worst cases;

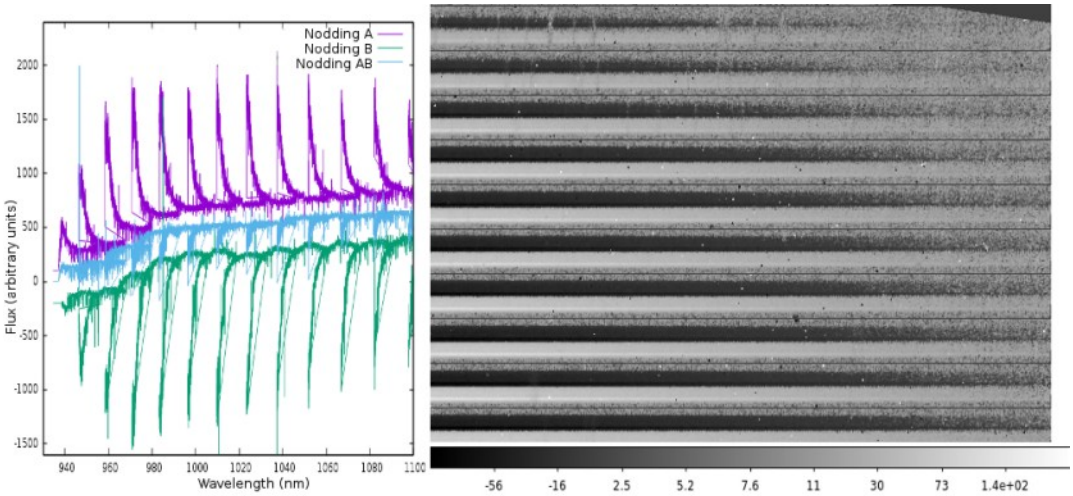

Figure 13: Behavior of the blue part of the blue orders for A, B and AB nodding and signal of the 2D straightened image.

- when the SNR is very low, it is possible to see a jump in the flux level in the middle of the orders: it is caused by the small differences between the detector's quadrants.

  Usually the nodding subtraction takes care of this effect, but in some rare cases it may remain. It can be manually removed by splitting the orders evenly in two and applying a linear normalization to one half with respect to the other.

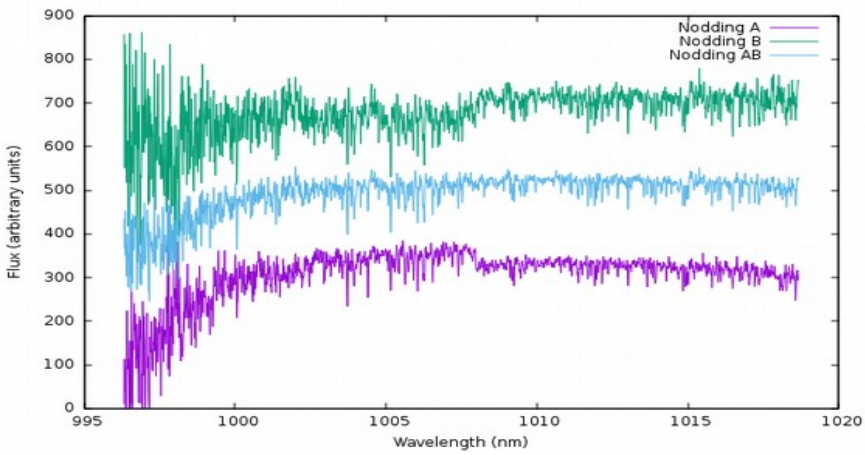

Figure 14: The jump in the middle of the orders is usually averaged out in the AB files.

This effect is usually visible only in the A and B noddings, as it tends to be averaged out in the combined AB spectra (see fig. 14);

- when the SNR is high, it is possible to see some fringing effects in the central/red part of the blue orders (see Fig. 15). This is caused by the detector itself (an HAWAII-2 from Rockwell Scientific): the sensitive part of the detector is beyond a 0.38 mm thick sapphire substrate that behaves as a Fabry-Pérot, generating an interference fringing with peak-to-peak distance approximately equal to $0.75 \cdot (wl/1000)^2$ nm.

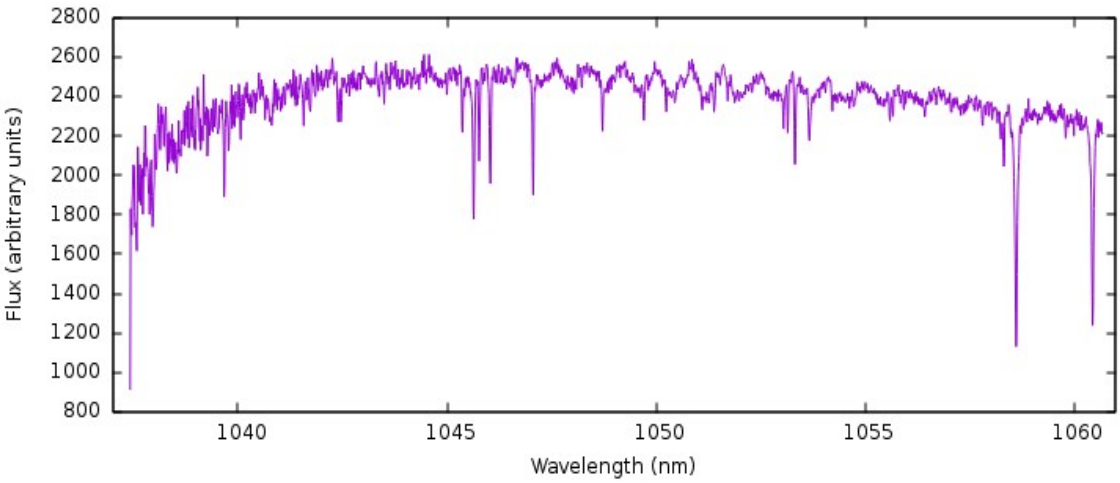

Figure 15: Fringing effect in a reduced spectrum (order 73).

Unfortunately this effect's amplitude is not constant, and as such it can not be removed a priori. Trying to model it and then remove it automatically will risk the removal of significant spectral information, as such it will not be done by GOFIO.

Figure 16 shows this fringing in a raw flat-field image: for visibility's sake, the effect is enhanced by subtracting the smoothed image from the original one.

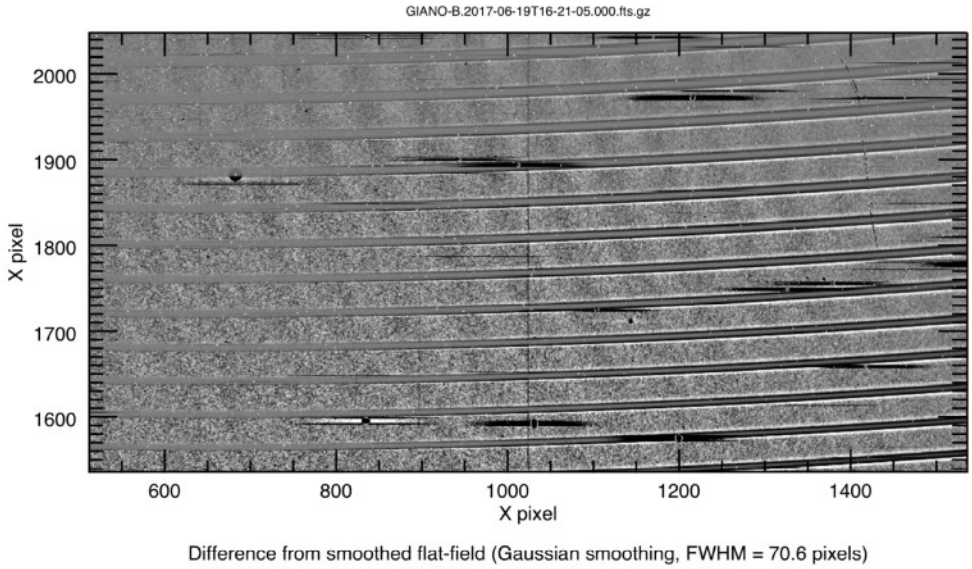

Figure 16: Fringing effect in the raw 2D data (flat-field). Figure courtesy of V. Andretta.

7 Appendix: drslib/varie.py

The functions that are used by at least two different reduction processes are defined in the file drslib/varie.py:

- badpix(): bad pixels removal. The frame is filtered using a rectangular filter 41-pixel wide and 1-pixel high and then the bad pixel mask is used to define the bad pixel positions and substitute their values with the filtered ones;
- stdcombine(): definition of the standard deviation weighted with RON and gain to be used for sigma-clipping flat-field and calibrations lamps before averaging them;
- shiftY(): automatically compute the vertical shift of the orders on the detector, if any.
- buildMaskC(): creation of the extraction mask from the straightened masterflat, the mask defines the exact position of the orders;
- optExtract(): optimal extraction and cosmic removal;
- extract(): extraction using a pre-defined extraction profile;
- UNe_linelist(): read the files with the U-Ne lines atlas;
- UNe_calibrate(): wavelength calibration using E. Oliva polynomials (default) or 3rd degree order polynomials;
- wcalib(): read the wavelength solution from a FITS header;
- rebin_linear(): rebin a spectrum with linear interpolation (not used, too slow);
- rebin2deg(): rebin a spectrum with parabolic interpolation (not used, too slow);
- rebin(): rebin a spectrum with linear interpolation;
- check_keyraw(): check for select keywords in the raw files, prompt for values if not found (not used anymore)
- check_keywords(): check for all the useful keywords in the raw files, prompt for values if not found (not used anymore);
- berv_corr(): compute barycentric correction (radial velocity and time);
- create_s1d(): create *_s1d output by rebinning the spectrum with a constant step in wavelength and merging the orders. To better merge the order, they are roughly normalized with a linear fit in the central region (default setting, it can be disabled in the configuration file);
- random_id(): create a unique ID for the reduced files for archiving purposes.

8 Appendix: drslib/config.py

The configuration file config.py contains two functions (to determine the date for the online reduction and to read the offline configuration file gofio.cfg) and the definitions of the CONFIG dictionary.

The CONFIG variables and their meanings are explained below. The values that must to be checked and eventually modified are shown in boldface. The values that may be modified are shown in italics.

- **Basic definitions**

CONFIG['APPNAME'] → name of the program (GOFIO) , do not change

CONFIG['VERSION'] → version of the program, do not change

CONFIG['DATE'] → the date for the online reduction, automatically determined

CONFIG['OFFLINE'] → defined with the default value False, do not change

- **Base directories**

CONFIG['TMP_DIR'] → where the temporary files are stored, do not change

CONFIG['APP_DIR'] → where GOFIO is located, do not change

CONFIG['BASE_RAW'] → where the raw yyyy-mm-dd directories are located

CONFIG['BASE_RED_DIR'] → where the reduced yyyy-mm-dd directories will be written

CONFIG['BASE_RAMP'] → where the ramp database is located (used only by the online version)

- **Nested directories**

CONFIG['RAW_DIR'] → the raw directory for the night (automatically created)

CONFIG['RED_DIR'] → the reduced directory for the night (automatically created)

CONFIG['OFFLINE_DIR'] → the offline reduced directory, defined as False by default

CONFIG['RED_CALIB'] → the night reduced calibration directory

CONFIG['RED_STR'] → the night intermediate straightened files directory

- **Resources directory and files, logger and straighten options**

CONFIG['RES_DIR'] → the resource directory

CONFIG['BADPIX_MASK'] → the filepath of the bad pixel mask

CONFIG['LOGGER'] → logger option

CONFIG['LOG_FILE'] → filepath of the night log file

CONFIG['STRAIGHT'] → filepath of the straightening fortran program

CONFIG['STRAIGHT_OPT'] → optional values for the straightening program

CONFIG['BASE_CALIB_DIR'] → base calibration directory

CONFIG['CALIB_DIR'] → calibration files directory, nested in CONFIG['BASE_CALIB_DIR']

CONFIG['CALIB_DB_DIR'] → calibration database directory, in CONFIG['BASE_CALIB_DIR']

CONFIG['WEBUI_DB_DIR'] → reduction database directory

- **Ouput suffixes**

CONFIG['MERGED'] → s1d

CONFIG['UNMERGED'] → ms1d

- **Keywords needed by GOFIO, they must be in the raw files header**

CONFIG['KEYS'] → {'OBJECT' : 'OBJECT', 'OBSTYPE' : 'OBS-TYPE', 'OBSMODE' : 'OBSMODE', 'IMANAME' : 'IMANAME', 'FILENAME' : 'FILENAME', 'NCOMBINE' : 'NCOMBINE', 'MJD' : 'MJD-OBS', 'EXPTIME' : 'EXPTIME', 'NODSTARE' : 'HIERARCH TNG TPL NAME', 'GROUPN' : 'GROUP_N', 'GROUPI' : 'GROUP_I', 'SLIT' : 'SLIT_POS', 'STARE' : 'HIERARCH TNG TPL STAREPOS', 'GROUPID' : 'HIERARCH TNG TPL GROUPID', 'ID' : 'HIERARCH TNG TPL OBSID', 'PID' : 'HIERARCH TNG TPL PAIRID', 'NPAIRS' : 'HIERARCH TNG TPL NPAIRS' , 'IPAIRS' : 'HIERARCH TNG TPL IPAIRS' 'NREP' : 'HIERARCH TNG CAT EXP NREP', 'EXTMODE' : 'HIERARCH TNG CAT DR SPEXTMODE', 'RA' : 'HIERARCH TNG TEL TARG ALPHA', 'DEC' : 'HIERARCH TNG TEL TARG DELTA', 'PMA' : 'HIERARCH TNG TEL TARG PMA', 'PMD' : 'HIERARCH TNG TEL TARG PMD', 'EQUINOX' : 'HIERARCH TNG TEL TARG EQUINOX', 'TARG_TYPE' : 'HIERARCH TNG CAT TARG TYPE', 'AM' : 'AIRMASS', 'RADVEL' : 'HIERARCH TNG TEL TARG RADVEL' }

- **Expected values of some raw keywords**

CONFIG['DARK'] → expected OBSTYPE value for dark images

CONFIG['FLAT'] → expected OBSTYPE value for flat-field images

CONFIG['WCAL_UNE'] → expected OBSTYPE value for U-Ne images

CONFIG['WCAL_FP'] → expected OBSTYPE value for Fabry-Pérot images

CONFIG['SCIENCE'] → expected OBSTYPE value for scientific images

CONFIG['NODVALUE'] → expected NODSTARE value for nodding

CONFIG['STAREVALUE'] → expected NODSTARE value for stare

CONFIG['A'] → expected SLIT value for A position

CONFIG['B'] → expected SLIT value for B position

CONFIG['C'] → expected SLIT value for C (stare) position

CONFIG['OBJ'] → expected STARE value for Object position

CONFIG['SKY'] → expected STARE value for Sky position

CONFIG['UNKNOWN'] → expected value for nodding observations

CONFIG['EXTPAIR'] → expected EXTMODE value for pair extraction

CONFIG['EXTAVG'] → expected EXTMODE value for group extraction

CONFIG['EXTDEFAULT'] → default extraction EXTMODE, if not found

CONFIG['TELLURIC'] → expected TARG_TYPE value for telluric stars (for RV computation)

CONFIG['FASTROT'] → expected TARG_TYPE value for fast-rotating stars (RV)

CONFIG['TYPE_SCIENCE'] → expected TARG_TYPE value for all other objects (RV)

- **Keywords written by GOFIO in the reduced spectra**

CONFIG['KEY_DRS'] → base keyword name

CONFIG['RON_EFF'] → effective RON

CONFIG['GAIN_EFF'] → effective gain

CONFIG['TEXP_EFF'] → effective exposure time

CONFIG['SPEC_USED'] → images combined to obtain the final result

CONFIG['SPEC_MJD'] → MJD of the combined images

CONFIG['SNR'] → SNR in the middle of the order

CONFIG['STRAIGHT_PAR'] → parameters used for straightening the orders

CONFIG['DRS_MJD'] → MJD of the combined image

CONFIG['BERV'] → barycentric correction (km/s)

CONFIG['HJD'] → heliocentric JD (UTC) in MJD

CONFIG['BJD'] → barycentric JD (TDB) in MJD

CONFIG['AIRMASS'] → airmass of the combined image

CONFIG['MASTERFLAT'] → masterflat used in the reduction

CONFIG['MASTERLAMP'] → masterlamp used for wavelength calibration

CONFIG['KEY_WEXT'] → extraction window for the default masterlamp (only in U-Ne/FP data)

CONFIG['DRS_VERSION'] → version number of GOFIO

CONFIG['RED'] → (for archive use) flag for a GOFIO reduced file

CONFIG['RED_TYPE'] → (for archive use) file type (DARK/FLAT/LAMP/SCIENCE/...)

CONFIG['RED_SUBTYPE'] → (for archive use) file subtype (specific reduction results)

CONFIG['RED_SLIT'] → slit position or combination (as in AB nodding results)

CONFIG['YPOS']/['AYPOS']/['BYPOS'] → pixel position of the maximum of the signal on the slit, respectively for stare images, nodding A images and nodding B images

CONFIG['YPOS_LOW']/['AYPOS_LOW']/['BYPOS_LOW'] → pixel position of the lower border of the extraction profile on the slit, respectively for stare images, nodding A images and nodding B images

CONFIG['YPOS_UP']/['AYPOS_UP'] /['BYPOS_UP'] → pixel position of the upper border of the extraction profile on the slit, respectively for stare images, nodding A images and nodding B images

- **Wavelength calibration keywords**

CONFIG['CAL_FUNC'] → polynomial used for the calibration (default: Oliva's polynomial)

CONFIG['CAL_FAILED'] → flag for failed wavelength calibration

CONFIG['WLFIT'] → keyword where the polynomial used is stored

CONFIG['WLFIT_FUNC'] → function of the polynomial used

CONFIG['WLCOEFFS'] → keywords where the polynomial coefficients, parameters and RMS are stored

- **Databases configuration**

CONFIG['RAMP_NAME'] → name of the ramp database

CONFIG['DB_RAMP'] → filepath of the ramp database

CONFIG['DB_RAMP_TBL'] → name of the table in the ramp database

CONFIG['DB_RAMP_COLS'] → names of the columns in the ramp database table

CONFIG['DB_CALIB'] → name of the calibration database

CONFIG['DB_CALIB_PATH'] → filepath of the calibration database

CONFIG['DB_CALIB_TBL'] → name of the table in the calibration database

CONFIG['DB_CALIB_COLS'] → name of the columns in the calibration database table

CONFIG['DB_CALIB_DATATYPE'] → datatype of the columns in the calibration database table

CONFIG['DB_ONLINE'] → online reduction database

CONFIG['DB_OFFLINE'] → offline reduction database

CONFIG['DB_ONLINE_PATH'] → filepath of the online reduction database

CONFIG['DB_OFFLINE_PATH'] → filepath of the offline reduction database

CONFIG['DB_2D_TBL']/['DB_1D_TBL']/['DB_CAL_TBL'] → names of the tables in the reduction database (online or offline)

CONFIG['DB_2D_COLS']/['DB_1D_COLS'/['DB_CAL_COLS'] → names of the columns in the reduction database tables (online or offline)

CONFIG['DB_2D_DATATYPE']/['DB_1D_DATATYPE']/['DB_CAL_DATATYPE'] → datatype of the columns in the reduction database tables (online or offline)

CONFIG['DB_NIGHT'] → reduction database (default: online database)

CONFIG['DB_NIGHT_PATH'] → filepath of the reduction database (default: online filepath)

- **Some information needed for the extraction process**

CONFIG['MASK_C'] → filepath of the extraction mask

CONFIG['A_POS'] → guess A position on the slit in pixel

CONFIG['B_POS'] → guess B position on the slit in pixel

CONFIG['C_POS'] → guess C position on the slit in pixel

CONFIG['Y_POS'] → maximum allowed difference between expected and real slit position

CONFIG['HWTM'] → HWTM limit for contamination between A and B noddings

CONFIG['S1D'] → flag for the creation of s1d output (default: True)

CONFIG['S1D_NORM'] → flag for normalization of the s1d output (default: False)

CONFIG['S1D_STEP'] → constant step to use in s1d spectra in nm

CONFIG['NCOSMIC'] → maximum number of cosmic to be removed in each order

- **Some information needed for the wavelength calibration**

CONFIG['WAVE_SELECT'] → filepath of the selected line atlas

CONFIG['WAVE_ALL'] → filepath of the global line atlas

CONFIG['WAVE_FIT'] → parameters used to validate the gaussian fit of the emission lines

CONFIG['XC_GUESS'] → initial values of x_c order by order

CONFIG['L0_GUESS'] → intial values of λ_0 order by order

- **Miscellaneous calibration configuration**

CONFIG['DO_CALIB'] → flags to reduce or not the calibrations or only the calibrations

CONFIG['USE_FLAT'] → flags to decide the division by the masterflat

CONFIG['DARKLIST'] → expected exposure times of dark frames

CONFIG['NDARK'] → number of dark frames expected for each dark exposure

CONFIG['WEXT'] → extraction half-window for the default wavelength calibration

- **Some information on the detector**

CONFIG['RON'] → read-out-noise

CONFIG['GAIN'] → gain

CONFIG['XCCD'] → pixel dimension on the x-axis

CONFIG['YCCD'] → pixel dimension on the y-axis

CONFIG['N_ORD'] → number of echelle orders on the detector

CONFIG['W_ORD'] → width of the straightened orders in pixel

CONFIG['SHIFT_Y'] → expected bottom position of the flat signal on the lowest order and on the left side of the detector (do not change)

- **Frames quality checks**

CONFIG['DARK_MEAN'] → upper limits for the mean values of the dark frames, depending on the exposure times and the quadrants

CONFIG['FLATSIGNAL'] → lower limit for the mean value of the flat to pass the quality check

CONFIG['SCIENCECHECK'] → region to use for SCIENCE images quality check

CONFIG['NODSIGNAL'] → lower limit for the mean value of the SCIENCE images in the region defined above to pass the quality check

- **Miscellaneous**

CONFIG['LEAP_UPDATE'] → used by baycorrpy to update the leap second. Default setting is True, but it must be set to False if the online updated table (http://maia.usno.navy.mil/ser7) is not available, otherwise the reduction process will slow down (*e.g.*, up to 900s per nodding).

In case the gofio.cfg file is found or the offline reduction is launched by command line, some configuration values are overwritten directly in drslib/gofio.py