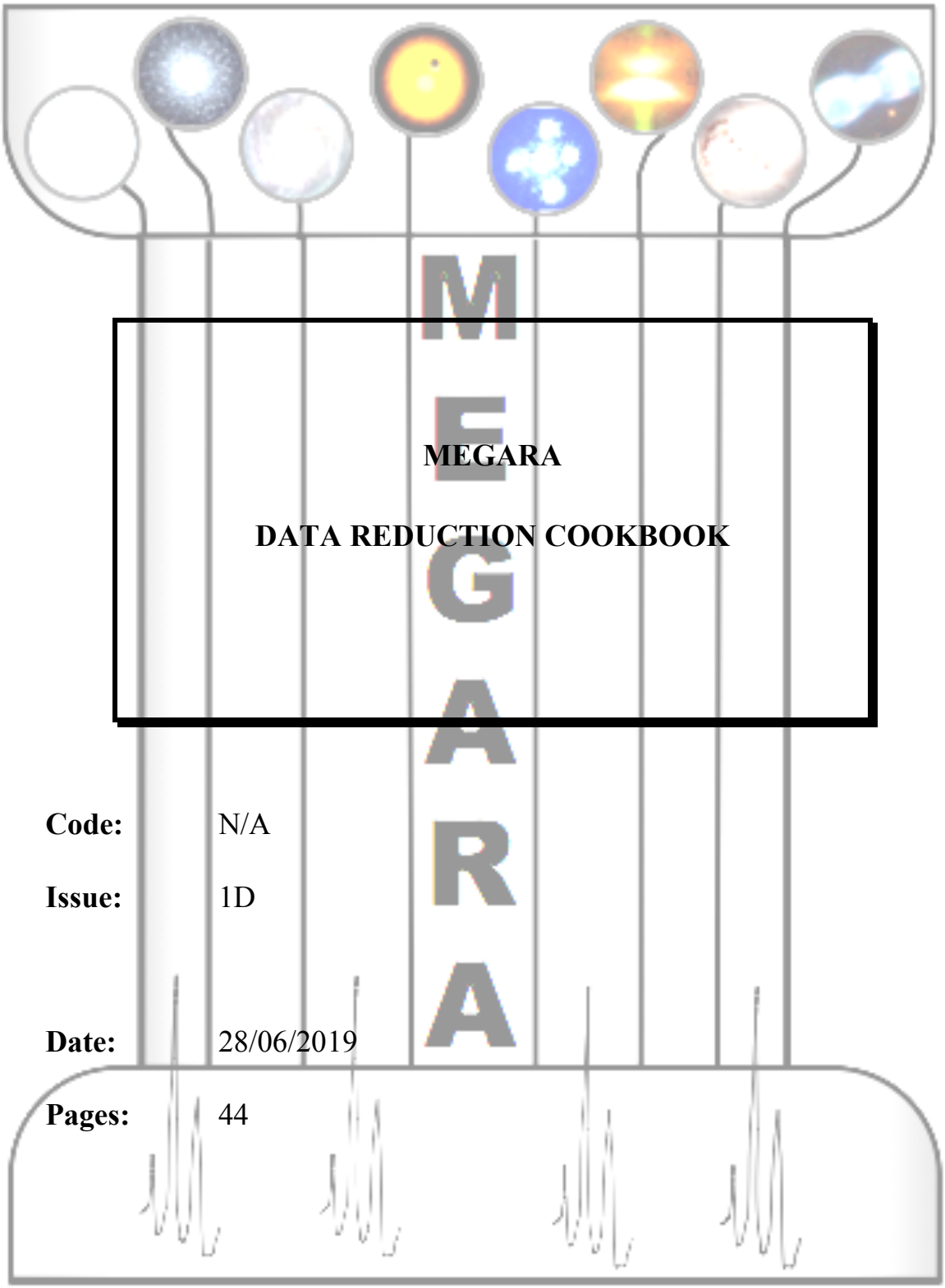
Universidad Complutense Madrid

GRAN TELESCOPIO CANARIAS GTC



MEGARA

DATA REDUCTION COOKBOOK

Code: N/A

Issue: 1D

Date: 28/06/2019

Pages: 44

Authors:	África Castillo Morales, Sergio Pascual Ramírez, Armando Gil de Paz
Revised by:	Armando Gil de Paz
Approved by:	Armando Gil de Paz

Distribution List:

Name	Affiliation	Date
MEGARA team	N/A	1/10/2018

Acronyms:

ADC	Analog-Digital Converter / Conversion
AIV	Assembly, Integration and Verification
CCD	Charged-Coupled Device
DRP	Data Reduction Pipeline
DU	Digital Unit
FC	Folded-Cassegrain
FMAT	Fiber-MOS Assignment Tool
FWHM	Full-Width at Half-Maximum
GTC	Gran Telescopio CANARIAS
ICM	Instrument Calibration Module
IFU	Integral Field Unit
IFS	Integral Field Spectrograph / Integral Field Spectroscopy
JSON	JavaScript Object Notation
LCB	Large Compact Bundle
LICA	Laboratorio de Instrumentación Científica Avanzada
MEGARA	Multi-Espectrógrafo en GTC de Alta Resolución para Astronomía
MOS	Multi-Object Spectrograph / Multi-Object Spectroscopy
PDF	Portable Document Format
RSS	Row-Stacked Spectrum
UCM	Universidad Complutense de Madrid

Change Control

Issue	Date	Section	Page	Change description
1A	3/9/2018			First issue
1B	1/10/2018			Second issue. Added: VPH table, info on how to update the DRP installation, how to visualize traces (overplot_traces) and FITS (numina-ximshow). The description of the *final_rss.fits* 92x4300 extension has been included.
1C	18/06/2019			Third issue. Minor problems when using the MEGARA DRP in Mac OS X are reported and solutions are provided. We also describe how to process without flux calibration neither atmospheric extinction correction.
1D	28/06/2019			MEGARA DRP is only compatible with Python 3.5 or later versions. Installation is now compatible with the conda 4.4 distributions. More details on the normalize_region & continuum_region requirements for the TwilightFiberFlat recipe are provided.

Reference Documents

Nº	Document Name	Code
R.1	MEGARA, the R=6000-20000 IFU and MOS of GTC	Proc. of the SPIE, 10702-42, 20 pp. (2018)
R.2	MEGARA, the new intermediate-resolution optical IFU and MOS for GTC: getting ready for the telescope	Proceedings of the SPIE, Volume 9908, id. 99081K, 20 pp. (2016).
R.3	First scientific observations with MEGARA at GTC	Proc. of the SPIE, 10702-43, 20 pp. (2018)

Reference Documents (GTC codes)

Nº	Document Name	Code

INDEX

1. INTRODUCTION

1.1 Scope

The goal of this document is to guide any potential user of the MEGARA instrument in its data processing, from the raw data provided by the GTC to wavelength- and flux-calibrated scientific-valid data. The MEGARA data processing described in this document will be done using the MEGARA Data Reduction Pipeline (DRP) which is available through *github* at https://github.com/guaix-ucm/. The different releases of this document will cope with any major change in the MEGARA DRP.

1.2 MEGARA instrument

MEGARA (*Multi-Espectrógrafo en GTC de Alta Resolución para Astronomía*) is a fiber-fed spectrograph with both Integral-Field (IFU) and Multi-Object (MOS) capabilities that was installed and commissioning at the 10.4m GTC telescope in the Spring of 2017. Since semester 2018B, MEGARA is available to the GTC community (Spain, Mexico and UF) in its two modes (LCB and MOS).

The MEGARA IFU, which is called Large Compact Bundle (LCB hereafter), covers a field of 12.5 x 11.3 arcsec^2 using 567 hexagonal spaxels of 0.62 arcsec in size plus 56 sky spaxels of equal size distributed in 8 bundles of 7 fibers distributed in the outskirts of the field at about 2 arcmin from the center of the LCB. The MOS makes use of a set of 92 robotic positioners each hosting a minibundle of 7 spaxels also of 0.62 arcsec in size each spaxel. These can patrol overlapping circular regions of 28 arcsec in diameter. These robotic positioners a distributed in a square region of 3.5 x 3.5 arcmin^2, which roughly corresponds to the flat and non-vignetted focal plane of GTC at its Folded-Cassegrain F (FC-F) focus (see ***Figure 1***). The MOS can be reconfigured starting from a list of target potions in matter of roughly a minute to a few minutes, depending on the level of number of overlapping patrol areas to be explored in a given configuration.

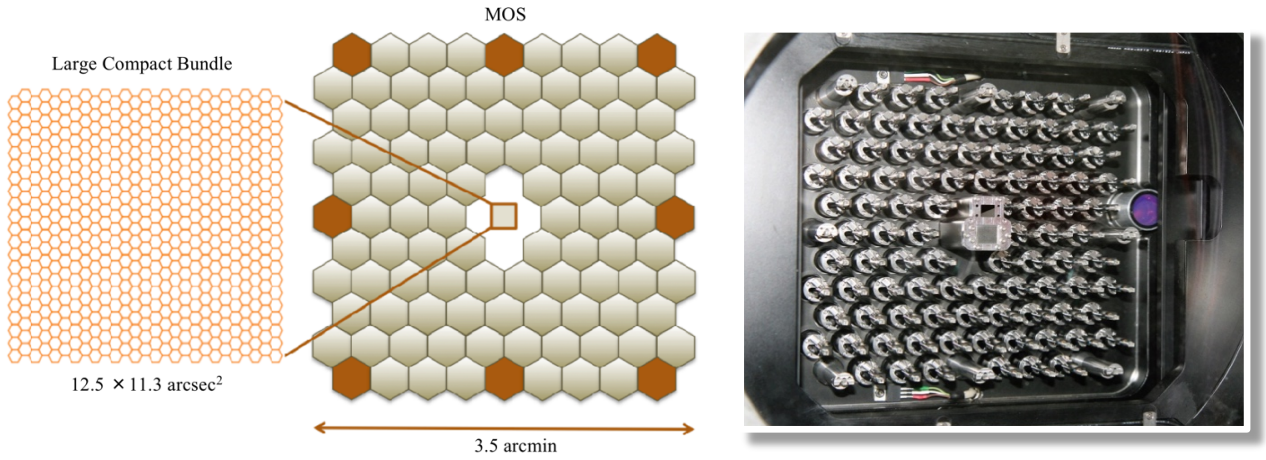


Figure 1: *LCB and MOS in the focal plane of MEGARA at the Folded-Cass F focus of GTC. Left: Layout of the monolithic microlens array of the LCB placed at the optical axis of the instrument. Center: Hexagons representing the patrol areas of the 92 robotic positioners of the MEGARA MOS (in light grey) along with the positions of the eight sky bundles that are mounted along the LCB pseudo-slit (in orange). Note that the actual patrol areas are overlapping circular regions of 28 arcsec in diameter, while the distance between adjacent positioners is 24.5 arcsec. Right: MEGARA focal plane before the field lens was installed at the Laboratorio de Instrumentación Científica Avanzada (LICA-UCM) laboratory.*

Both the LCB and the MOS along with other subsystems (focal-plane cover, Folded-Cassegrain rotator adapter, etc.) are located at the FC-F focus of GTC. The 623 (567+56) fibers of the LCB and the 644 fibers of the 92 robotic positioners of the MOS are routed through from the FC-F rotator to the Nasmyth A platform following a 44.5m-long path until they reach the MEGARA spectrograph. The MEGARA spectrograph is a fixed-angle (68º) collimator-camera system that is fed by two interchangeable curved pseudo-slits (LCB/MOS). The collimator is an all-refractive F/3 system composed by 5 lenses (1 aspheric singlet and 2 doublets) while the also all-refractive camera is composed by 7 lenses (two doublets, one with a CaF_2 lens, and 3 singlets). In between collimator and camera, the spectrograph pupil can host different types of Volume-Phase Holographic (VPH) disperser elements, namely the low- (LR), mid- (MR), and high-resolution (HR) VPHs. Six LR VPHs cover the entire optical window at R=6,000, while 10 MR VPHs provide also full optical coverage but at R=12,000. Finally, the two HR VPHs allow observing in the Hα+[NII] region and in the CaT region with R=20,000, although the optical design could in principle accommodate HR VPHs at any other optical wavelength. In ***Figure 2*** we show the resolving power and spectral coverage for each VPH as measured during the integration and commissioning of the instrument[1].

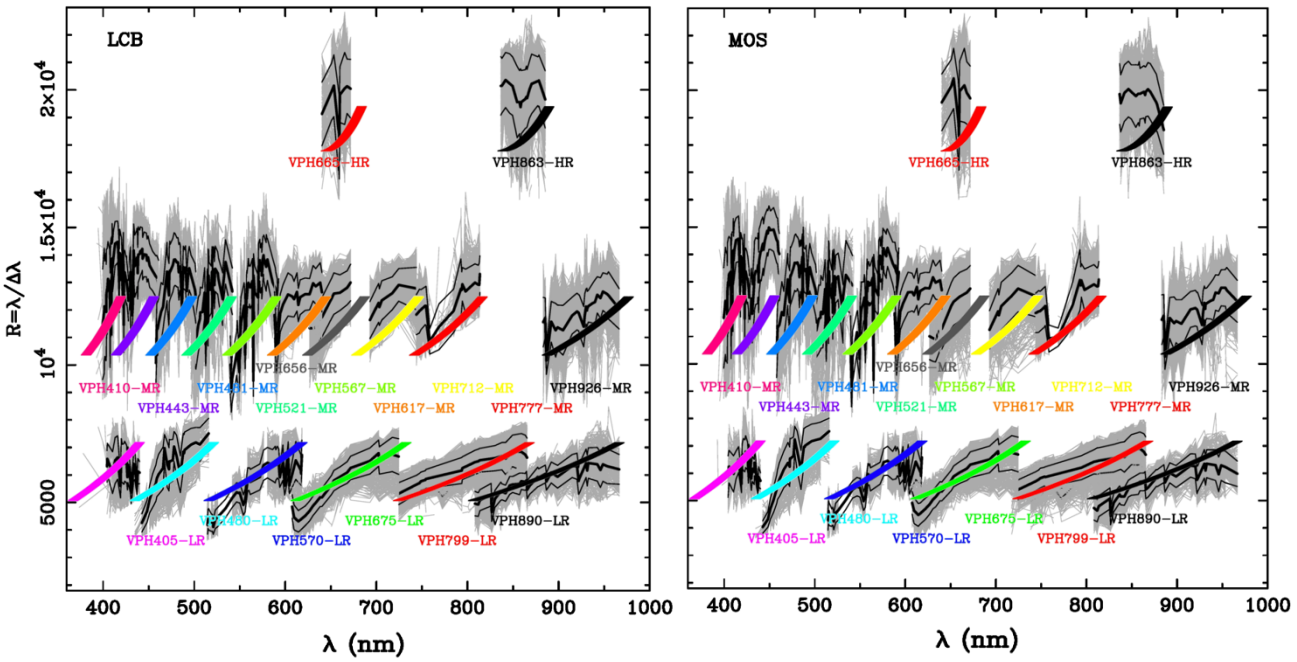


Figure 2: *Plots showing the relation between resolving power (RFWHM) and wavelength coverage for all 18 MEGARA VPHs and for the LCB (left) and MOS (right) modes. Design values (colored lines) and measurements (grey lines that correspond to individual fiber spectra, while black thick and thin lines represent the mean and mean±1σ curves when all fiber spectra are used) are both shown.*

The details on the different VPHs that can be used with MEGARA is given in Table 1. This table also includes the reciprocal (linear) dispersion (CDELT) and wavelength for the initial pixel (CRVAL for CRPIX=1) as

[1] Note that in some of the cases the spectral coverage shown is shorter than the one actually achieved simply because the spectral lamp lacks bright spectral features (on which to measure the spectral resolution and resolving power), especially at the blue end of the optical spectral range.

adopted for the MEGARA DRP for different VPHs. The user is referred to different publications to learn more about the MEGARA instrument, including [R.1], [R.2] and [R.3].

VPH Name	Setup	R_{FWHM}	λ_1-λ_2 Å	λc Å	$\Delta\lambda$ (@ λc) Å	Δv km/s	lin res Å/pix	λ(pix1) Å
VPH405-LR	LR-U	6028	3653 – 4386	4051	0.672	50	0.186	3620
VPH480-LR	LR-B	6059	4332 – 5196	4800	0.792	49	0.23	4280
VPH570-LR	LR-V	6080	5143 – 6164	5695	0.937	49	0.27	5060
VPH675-LR	LR-R	6099	6094 – 7300	6747	1.106	49	0.31	6030
VPH799-LR	LR-I	6110	7220 – 8646	7991	1.308	49	0.37	7140
VPH890-LR	LR-Z	6117	8043 - 9630	8900	1.455	49	0.41	7960
VPH410-MR	MR-U	12602	3917 - 4277	4104	0.326	24	0.089	3905
VPH443-MR	MR-UB	12370	4225 – 4621	4431	0.358	24	0.10	4210
VPH481-MR	MR-B	12178	4586 – 5024	4814	0.395	25	0.11	4568
VPH521-MR	MR-G	12035	4963 – 5443	5213	0.433	25	0.122	4944
VPH567-MR	MR-V	11916	5393 – 5919	5667	0.476	25	0.132	5375
VPH617-MR	MR-VR	11825	5869 – 6447	6170	0.522	25	0.145	5850
VPH656-MR	MR-R	11768	6241 – 6859	6563	0.558	25	0.16	6210
VPH712-MR	MR-RI	11707	6764 – 7437	7115	0.608	26	0.17	6735
VPH777-MR	MR-I	11654	7382 – 8120	7767	0.666	26	0.1845	7360
VPH926-MR	MR-Z	11638	8800 - 9686	9262	0.796	26	0.225	8770
VPH665-HR	HR-R	18700	6445 - 6837	6646	0.355	16	0.0974	6390
VPH863-HR	HR-I	18701	8372 - 8882	8634	0.462	16	0.13	8350

Table 1: *MEGARA VPHs: scientific requirements (The resolution, $R_{FWHM}=\lambda/\Delta\lambda_{FWHM}$, is derived from the FWHM ($\Delta\lambda_{FWHM}$) of the 1D spectra). The values of the linear reciprocal dispersion and the wavelength of pixel 1 correspond to the linear solution implemented in the MEGARA DRP.*

Note that the reciprocal dispersion is the one used for the linear solution in the images processed by the MEGARA Data Reduction Pipeline.

2. MEGARA DATA REDUCTION PIPELINE

The deployment of the MEGARA instrument at GTC was accompanied by the installation of a fully functioning Data Reduction Pipeline (DRP hereafter) developed in Python that worked both online at the telescope and offline. The online version of the DRP allows for on-the-fly data processing, which includes bias correction, trimming, fiber tracing and fixed-aperture extraction, fiber-flat and twilight-flat correction and wavelength calibration. The offline processing (to which this cookbook is devoted) additionally includes a detailed cross-talk-corrected extraction and absolute flux calibration whenever possible. The MEGARA DRP is distributed by GTC at http://www.gtc.iac.es/instruments/megara/config/megaradrp-0.6.dev2.tar.gz. Updates to the DRP can be obtained through *github* at https://github.com/guaix-ucm/megaradrp (see also section 3 below)

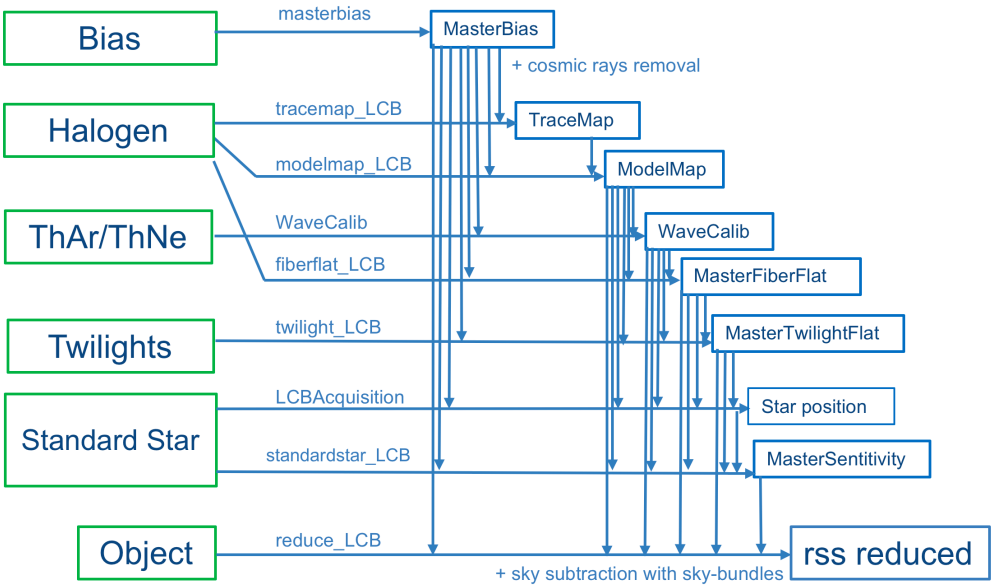


Figure 3: *Data processing scheme of the MEGARA DRP.*

The MEGARA DRP has been designed to cope with all effects associated to the observation with a fiber-fed spectrograph on which the detection of the light is done with a Charge-Coupled Device (CCD). These effects include the removal of the bias level and the dark current associated to the MEGARA CCD, the tracing and extraction of the flux from each fiber on the CCD, the variation in the wavelength calibration solution along the (pseudo-)slit of the spectrograph and the correction from the variation in sensitivity (from blue-to-red and global) from fiber to fiber and the determination of the system efficiency. In ***Figure 3*** we show the data processing scheme followed by the MEGARA DRP. We note here that the wavelength calibration is performed quite early on in the reduction procedure as the correction for blue-to-red variation in sensitivity has to be done once the wavelength of the light falling in each pixel and each fiber is known.

The final products of the MEGARA DRP are "reduced" Row-Stacked Spectra (RSS hereafter) 2D images including for 623 (644) fiber spectra for the LCB (MOS) mode, all using a common flux calibration and wavelength solution with constant reciprocal dispersion for all fibers. Based on the averaged spectrum of all fibers to be used for sky subtraction (by default all 56 sky fibers in the LCB and all unassigned minibundles in the case of the MOS) the DRP also generates a sky-subtracted "final" RSS spectrum. No combo products combining different spectral setups are yet generated.

3. DRP INSTALLATION

The MEGARA pipeline is a Python package, for Python 3.5 or greater.

The easiest method of installing megaradrp is using prebuilt packages. You can also build from source or directly from the development version. All the commands in the following sections are to be run under **bash shell**. More details are in the MEGARA DRP *readthedocs* documentation[2].

Suggestion: what method of installation should I use?

* If you are already familiar with one method, use it (conda or virtualenv), since both a fully supported.

* In macOS, there is a well-known compatibility problem between virtualenv and matplotlib[3], so we recommend setting up conda.

* In Linux, virtualenv is easier to setup

3.1 Install in virtualenv

Virtualenv[4] is a tool that allows to create isolated Python environments. There is also a module in the standard library called `venv` with roughly the same functionallity.

The steps to run MEGARA DRP in a virtual environment are:

3.1.1 Create a virtual environment using either virtualenv or venv.

In order to create a virtual environment called e.g. megara using `venv`:

```
bash-3.2$ python3 -m venv megara /path/to/
```

With `virtualenv`:

```
bash-3.2$ virtualenv-3 megara /path/to/
```

The directory `/path/to` represents the location of the environment. It can be any valid directory path, even the local directory `.` .

3.1.2 Activate the environment.

After creating the environment, the directory `/path/to/megara` contains a Python tree. One of the directories is `/path/to/megara/bin`, which contains a script called `activate`. To activate the environment, we `source` (a bash shell command) this script file:

[2] https://megara-drp.readthedocs.io/en/latest/installation.html

[3] https://matplotlib.org/faq/osx_framework.html

[4] https://virtualenv.pypa.io/en/stable/installation/

```
bash-3.2$ source /path/to/megara/bin/activate
```

which yields a different system prompt to the user:

```
(megara) bash-3.2$
```

Now, the name of the environment appears before the standard prompt. We can use the environment only on those consoles / terminals where we have previously activated it.

3.1.3 Install megaradrp with pip

After the activation, we can install megaradrp with `pip`. This is the standard Python tool for package management. It will download the package and its dependencies, unpack everything and compile when needed.

What follows is a sample of the output:

```
(megara) bash-3.2$ pip install megaradrp

   Collecting megaradrp

   Collecting scikit-image (from megaradrp)

   Downloading
https://files.pythonhosted.org/packages/11/c7/ee75c79dcce057a3475763d611ec044737a708eaf5cc53426b0117
795ddb/scikit_image-0.14.0-cp35-cp35mu-manylinux1_x86_64.whl (25.4MB)

   Collecting scipy (from megaradrp)

   (...)

   Building wheels for collected packages: toolz, scandir

   Running setup.py bdist_wheel for toolz ... done

   Running setup.py bdist_wheel for scandir ... done

   Successfully built toolz scandir

   Installing collected packages: decorator, networkx, cloudpickle, numpy, toolz, dask, six,
PyWavelets, python-dateutil, subprocess32, cycler, backports.functools-lru-cache, pytz, pyparsing,
kiwisolver, matplotlib, scipy, pillow, scikit-image, enum34, atomicwrites, more-itertools, pluggy,
attrs, scandir, pathlib2, py, funcsigs, pytest, astropy, PyYaml, numina, megaradrp
```

3.1.4 Test the installation.

Now we can test the installation by running the `numina` command:

```
(megara) bash-3.2$ numina
```

```
DEBUG: Numina simple recipe runner version 0.17.3
```

3.1.5 Update within the environment

In order to update the MEGARA DRP in a virtualenv installation the user should execute:

```
(megara) bash-3.2$ pip install -U megaradrp
```

3.1.6 Deactivate the environment.

To exit the environment is enough to exit the terminal or run the command `deactivate`.

```
(megara) bash-3.2$ deactivate

bash-3.2$
```

3.2 Install in conda

Conda[5] was created with a target similar to `virtualenv`, but know has extended its functionallity to package management for different languages.

You can install miniconda[6] or anaconda[7]. The difference is that miniconda provides a light-weight environment and anaconda comes with lots of Python packages.

If you have updated the $PATH variable during install, you can call conda commands directly in the shell, like this:

```
bash-3.2$ conda info
```

If not, you will need to add the path to the command (an example path could be `miniconda3/bin`), like:

```
bash-3.2$ /path/to/conda/bin/conda info
```

If that is the case, you should add that path every time you run a conda command hereafter. Alternatively, you can initialize conda for your own shell by doing:

```
bash-3.2$ conda init bash
```

[5] https://conda.io/docs/

[6] See installation instructions at https://conda.io/miniconda.html

[7] See installation instructions at https://docs.anaconda.com/anaconda/install/

This works as it is if you are using a login-shell (terminal), but if you are using a xterm, you might also need to do:

`bash-3.2$ cp ~/.bash_profile ~/.bashrc` (do a backup copy of `~/.bashrc` if you have one already),

and open a new terminal/xterm. Below we will write the commands without the full path, for simplicity. Once conda is installed according to the instructions above, the steps to run MEGARA DRP under conda are (note that after conda 4.4 you will be already in the conda *(base)* environment):

3.2.1 Create a conda environment

We first recommend that you update your conda installation to its latest by doing:

```
(base) bash-3.2$ conda update conda
```

With conda, environments are created in a centralised manner (under directory `./envs` in your conda tree), we do not pass the path to the environment.

```
(base) bash-3.2$ conda create --name megara python=3
```

3.2.2 Install megaradrp with conda

Packages can be installed before activating the environment. We provide conda packages for megaradrp in the conda-forge channel[8]:

```
(base) bash-3.2$ conda install --name megara -c conda-forge megaradrp
   Fetching package metadata ..............
   Solving package specifications: .

   Package plan for installation in environment /home/spr/devel/miniconda3/envs/megara:
    The following NEW packages will be INSTALLED:
       astropy:                        2.0.8-py35_0          conda-forge
       atomicwrites:                   1.1.5-py35_0          conda-forge
       attrs:                          18.1.0-py_1           conda-forge
       ....
       ....
       ....
       zlib:                           1.2.11-h470a237_3     conda-forge
Proceed ([y]/n)? y
```

[8] https://conda-forge.org/

3.2.3 Activate the environment

The functionality is similar to virtualenv:

```
(base) bash-3.2$ conda activate megara

(megara) bash-3.2$
```

Again, after activating the environment, the name of the environment appears before the standard prompt. We can use the environment only on those consoles / terminals where we have previously activated it.

3.2.4 Test the installation

Now we can test the installation by running the `numina` command:

```
(megara) bash-3.2$ numina

   DEBUG: Numina simple recipe runner version 0.17.3
```

3.2.5 Update within the environment

In order to update the MEGARA DRP within the conda environment the user should execute:

```
(megara) bash-3.2$ conda update megaradrp
```

3.2.6 Deactivate the environment

To exit the environment is enough to exit the terminal or run the command `source deactivate`

```
(megara) bash-3.2$ conda deactivate

(base) bash-3.2$
```

3.2.7 Update outside the environment

Once outside the conda environment one can also update the MEGARA DRP installation by doing:

```
(base) bash-3.2$ conda update megaradrp -n megara
```

If you want to deactivate the conda *(base)* environment entirely you can run again:

```
(base) bash-3.2$ conda deactivate

bash-3.2$
```

4. DATA DESCRIPTION

In order to help the user in understanding the different execution steps of the MEGARA DRP, we describe in this section the characteristics of the main products, including input raw images and pipeline products (images and tables).

4.1 Raw Data

Raw data includes all FITS frames delivered to the user by GTC. These FITS images are 4196 x 4212 pixels in size have two extensions, the first one including the data themselves and the second one providing all the information about the fibers (positions in the sky, bundle to which they belong and whether they are devoted to the observation of target or sky). Among these images one can find bias frames (as they are obtained with the MegaraBiasImage observing mode they include the name of this mode it their filename), fiber-flat images (obtained with either the MegaraTraceMap or the MegaraFiberFlatImage observing modes), ThAr or ThNe HCL lamp spectra (obtained with the MegaraArcCalibration observing mode) and scientific observations with either the LCB (MegaraLcbImage or MegaraLcbAcquisition; this latter mode is commonly used when the target is a bright star, normally a spectrophotometric standard star) or the MOS (MegaraMosImage).

4.2 Pipeline Products

There are multiple types of products generated by the MEGARA DRP although they can be grouped in full-frame FITS images of 4096 x 4112 pixels in size (after the overscan+prescan regions are removed from the raw images), RSS images of 4300 x 623 (for LCB) or 4300 x 644 (for MOS) pixels, and structured data, which is in most cases are given in files of JSON format. Below we list the different products within these three groups along with the recipe that generates them:

- Full-frame FITS image products:
  - ***master_bias.fits*** (MasterBiasImage): Final image of the MasterBiasImage recipe.
  - ***reduced_image.fits*** (MegaraDarkImage, MasterTraceMap, MegaraModelMap, MegaraFiberFlatImage, MegaraArcCalibration, MegaraTwilightFlatImage, MegaraLcbStdStar, MegaraLcbAcquisition, MegaraLcbImage, MegaraMosImage, MegaraArcCalibration): Final image after all individual exposures have been processed and combined.
  - ***master_slitflat.fits*** (MegaraSlitFlat): Image obtained by observing a continuum-lamp light with the spectrograph out of its optimal focus. The level of de-focusing should be enough to ensure a uniform illumination through the entire CCD but keeping the wavelength of the light approximately the same at each given pixel that when the instrument is well focused.
  - ***fwhm_image.fits*** (MegaraArcCalibration): Voronoi map of the FWHM derived from the fits to the Gaussian profiles of all spectral lines identified in the arc-lamp image.
- RSS FITS image products:
  - ***master_fiberflat.fits*** (MegaraFiberFlatImage): Image to be applied to correct for variations in sensitivity in between fibers and from blue-to-red within each fiber.
  - ***master_twilightflat.fits*** (MegaraTwilightFlatImage): Image to be applied to correct for the effect of illumination introduced by the fiber-flat image when this was obtained through the FC-F ICM and differences between the pupil of the ICM and the GTC pupil when the object was observed. The values of the twilight-flat image are identical for all wavelengths but different from fiber to fiber (blue-to-red sensitivity variations were corrected with the fiber-flat image).

- ***reduced_rss.fits*** (MegaraLcbAcquisition, MegaraLcbStdStar, MegaraLcbImage, MegaraMosImage, MegaraArcCalibration): Processed image prior to the subtraction of the sky spectrum.
- ***sky_rss.fits*** (MegaraLcbAcquisition, MegaraLcbStdStar, MegaraLcbImage, MegaraMosImage, MegaraArcCalibration): RSS image showing signal only in the valid sky fibers. All other pixels are set to zero.
- ***final_rss.fits*** (MegaraLcbAcquisition, MegaraLcbStdStar, MegaraLcbImage, MegaraMosImage, MegaraArcCalibration): Processed image after the subtraction of the sky spectrum is performed. In the case of the MOS, this image includes an extension of 92 rows by 4300 columns where all 7 fibers of each minibundle have been added together.

- Structured products:
  - ***master_wlcalib.json*** (MegaraArcCalibration): File with the information on the wavelength calibration solution for every fiber.
  - ***master_traces.json*** (MasterTraceMap): File with the tracing information.
  - ***master_model.json*** (MasterModelMap): File with the information on how to account for the cross-talk between adjacent fibers in the detector.

In the case of the MegaraLcbStdStar recipe, the MEGARA DRP also generates two different 1D spectra, that of the standard star obtained after extracting the 37 spaxels around the centroid identified in the observation-result file (*star_spectrum.fits*) and also the resulting sensitivity function (*master_sentivity.fits*).

In addition to all these files, the results directory of every recipe includes also a file named *task.yaml* (file with the description of the execution of the recipe), the file *result.yaml* (names of the files resulting from the recipe and some quality-control information) and the *processing.log* logging file.

Besides ds9/SAOimage or similar software packages, the FITS products generated by the MEGARA DRP can be also visualized using the tool `numina-ximshow` distributed as part of **numina**.

5. DATA REDUCTION COOKBOOK

5.1 Getting Started

The first step to start the data reduction is to activate your environment (see details in Section 3.1 and 3.2):

5.2 Data organization

MEGARA DRP uses its own data organization to work. We need a directory named `MEGARA`, in our example this directory is under `data_reduction/`:

```
(megara) bash-3.2$ pwd

/Users/acm/Desktop/data_reduction/MEGARA
```

Under the `MEGARA/` directory we need to have the calibration tree with the specific name `ca3558e3-e50d-4bbc-86bd-da50a0998a48/`, which is the string that uniquely identifies the instrument configuration (a different name

was for example used during laboratory integration at LICA-UCM). Under the `MEGARA/` directory we can also have the requirements file named *control.yaml* needed to run the pipeline (see section 5.3; note that the *tree* command shown below might not be available in certain unix distributions; use "ls" instead).

```
(megara) bash-3.2$ tree -L 2
.
└── MEGARA
    ├── M15
    ├── M71
    ├── ca3558e3-e50d-4bbc-86bd-da50a0998a48
    └── control.yaml
```

The requirements file *control.yaml* contains the path for your `MEGARA/` directory:

```
rootdir: /Users/acm/Desktop/data_reduction
```

and useful information for performing the wavelength calibration of each VPH, including the number of emission lines, wavelength ranges and degree of polynomial fit to be used by the wavelength calibration recipe. In this file you can also specify the name for the extinction curve file used for the flux calibration recipe. This is simply an ASCII file with two space-separated columns, one with the wavelength in Angstroms and another with the magnitudes of extinction per unit airmass at the corresponding wavelength, i.e. the same format used for extinction curves within IRAF. We strongly recommend to use the standard extinction curve of the Roque de los Muchachos Observatory.

```
(megara) bash-3.2$ more control.yaml
version: 1
rootdir: /Users/acm/Desktop/REDUCTION_MEGARA/reduction_GTC_com
products:
  MEGARA:
   - {id: 2, type: 'ReferenceExtinctionTable', tags: {}, content: 'extinction_LP.txt'}
requirements:
  MEGARA:
    default:
      MegaraArcCalibration:
      - {name: nlines, tags: {vph: LR-U, speclamp: ThAr, insmode: LCB}, content: [25,25]}
      - {name: nlines, tags: {vph: LR-U, speclamp: ThAr, insmode: MOS}, content: [25,25]}
      - {name: nlines, tags: {vph: LR-B, speclamp: ThAr, insmode: LCB}, content: [10,10,15,5]}
      - {name: nlines, tags: {vph: LR-B, speclamp: ThAr, insmode: MOS}, content: [10,10,15,5]}
      - {name: nlines, tags: {vph: LR-V, speclamp: ThAr, insmode: LCB}, content: [15,5,10,7]}
      - {name: nlines, tags: {vph: LR-V, speclamp: ThAr, insmode: MOS}, content: [15,5,10,7]}
      - {name: nlines, tags: {vph: LR-R, speclamp: ThAr, insmode: LCB}, content: [14,7]}
      - {name: nlines, tags: {vph: LR-R, speclamp: ThAr, insmode: MOS}, content: [14,7]}
      - {name: nlines, tags: {vph: LR-I, speclamp: ThAr, insmode: LCB}, content: [14]}
      - {name: nlines, tags: {vph: LR-I, speclamp: ThAr, insmode: MOS}, content: [14]}
      - {name: nlines, tags: {vph: LR-Z, speclamp: ThNe, insmode: LCB}, content: [14,9]}
      - {name: nlines, tags: {vph: LR-Z, speclamp: ThNe, insmode: MOS}, content: [14,9]}
      - {name: nlines, tags: {vph: MR-U, speclamp: ThAr, insmode: LCB}, content: [8,10]}
      - {name: nlines, tags: {vph: MR-U, speclamp: ThAr, insmode: MOS}, content: [8,10]}
      - {name: nlines, tags: {vph: MR-UB, speclamp: ThAr, insmode: LCB}, content: [20]}
      - {name: nlines, tags: {vph: MR-UB, speclamp: ThAr, insmode: MOS}, content: [20]}
      - {name: nlines, tags: {vph: MR-B, speclamp: ThAr, insmode: LCB}, content: [11]}
      - {name: nlines, tags: {vph: MR-B, speclamp: ThAr, insmode: MOS}, content: [11]}
      - {name: nlines, tags: {vph: MR-G, speclamp: ThAr, insmode: LCB}, content: [10,10,8]}
      - {name: nlines, tags: {vph: MR-G, speclamp: ThAr, insmode: MOS}, content: [10,10,8]}
      - {name: nlines, tags: {vph: MR-V, speclamp: ThAr, insmode: LCB}, content: [13,8]}
      - {name: nlines, tags: {vph: MR-V, speclamp: ThAr, insmode: MOS}, content: [13,8]}
      - {name: nlines, tags: {vph: MR-VR, speclamp: ThNe, insmode: LCB}, content: [14]}
      - {name: nlines, tags: {vph: MR-VR, speclamp: ThNe, insmode: MOS}, content: [14]}
      - {name: nlines, tags: {vph: MR-R, speclamp: ThNe, insmode: LCB}, content: [9]}
      - {name: nlines, tags: {vph: MR-R, speclamp: ThNe, insmode: MOS}, content: [9]}
      - {name: nlines, tags: {vph: MR-RI, speclamp: ThNe, insmode: LCB}, content: [7]}
      - {name: nlines, tags: {vph: MR-RI, speclamp: ThNe, insmode: MOS}, content: [7]}
      - {name: nlines, tags: {vph: MR-I, speclamp: ThNe, insmode: LCB}, content: [5,5,5]}
      - {name: nlines, tags: {vph: MR-I, speclamp: ThNe, insmode: MOS}, content: [5,5,5]}
      - {name: nlines, tags: {vph: MR-Z, speclamp: ThNe, insmode: LCB}, content: [4,5,3]}
      - {name: nlines, tags: {vph: MR-Z, speclamp: ThNe, insmode: MOS}, content: [4,5,3]}
```

```
- {name: nlines, tags: {vph: HR-R, speclamp: ThNe, insmode: LCB}, content: [5]}
- {name: nlines, tags: {vph: HR-R, speclamp: ThNe, insmode: MOS}, content: [5]}
- {name: nlines, tags: {vph: HR-I, speclamp: ThNe, insmode: LCB}, content: [10]}
- {name: nlines, tags: {vph: HR-I, speclamp: ThNe, insmode: MOS}, content: [10]}
- {name: polynomial_degree, tags: {vph: LR-U, speclamp: ThAr}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: LR-B, speclamp: ThAr}, content: [5,5]}
- {name: polynomial_degree, tags: {vph: LR-V, speclamp: ThAr}, content: [5,5]}
- {name: polynomial_degree, tags: {vph: LR-R, speclamp: ThAr}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: LR-I, speclamp: ThAr}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: LR-Z, speclamp: ThNe}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: MR-U, speclamp: ThAr}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: MR-UB, speclamp: ThAr}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: MR-B, speclamp: ThAr}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: MR-G, speclamp: ThAr}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: MR-V, speclamp: ThAr}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: MR-VR, speclamp: ThNe}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: MR-R, speclamp: ThNe}, content: [3,3]}
- {name: polynomial_degree, tags: {vph: MR-RI, speclamp: ThNe}, content: [3,3]}
- {name: polynomial_degree, tags: {vph: MR-I, speclamp: ThNe}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: MR-Z, speclamp: ThNe}, content: [3,3]}
- {name: polynomial_degree, tags: {vph: HR-R, speclamp: ThNe}, content: [3,5]}
- {name: polynomial_degree, tags: {vph: HR-I, speclamp: ThNe}, content: [3,5]}
```

Another fundamental function of the calibration tree (`ca3558e3-e50d-4bbc-86bd-da50a0998a48/`) is to host the calibration products that will be used by the corresponding recipes, such as the MasterBias, MasterFiberFlat, MasterSensitivity, etc. Thus, once the files for these calibrations are generated, they should be copied under this calibration tree according to the following structure:

```
(megara) bash-3.2$ tree ca3558e3-e50d-4bbc-86bd-da50a0998a48/ -L 2
ca3558e3-e50d-4bbc-86bd-da50a0998a48/
├── LinesCatalog
│   ├── ThAr
│   └── ThNe
├── MasterBPM
│   └── master_bpm.fits
├── MasterBias
│   └── master_bias.fits
├── MasterFiberFlat
│   ├── LCB
│   └── MOS
├── MasterSensitivity
│   ├── LCB
│   └── MOS
├── MasterSlitFlat
│
├── MasterTwilightFlat
│   ├── LCB
│
├── ModelMap
│   ├── LCB
│   └── MOS
├── TraceMap
│   ├── LCB
│   └── MOS
└── WavelengthCalibration
    ├── LCB
    └── MOS
```

The content for the `LinesCatalog/` is specific for each VPH. In the following example the calibration files for the HR-R (LCB observing mode) and LR-R (MOS observing mode) VPHs are shown. When other VPHs are used, the user just needs to create the corresponding folders. It is recommended to have only one file in each calibration directory. For example, for the same VPH you can have several *master_traces.json* files with the information to trace the fibers light through the detector at the same day but at different ambient temperatures. Different files can be stored at the same directory, but the DRP is going to use the first file it encounters in alphabetical order. The user can name the desired file with prefix *"00_"* (e.g. *00_master_traces.json*) to be

sure this is the file to be used by the DRP. Note that the sorting of files named *"00_"* and *"000_"* might be different for the operative system and for the MEGARA DRP, so avoid making abusive use of these prefixes.

```
(megara) bash-3.2$ tree ca3558e3-e50d-4bbc-86bd-da50a0998a48/ -L 4
ca3558e3-e50d-4bbc-86bd-da50a0998a48/
├── LinesCatalog
│   ├── ThAr
│   │   ├── LR-R
│   │   │   └── LR-R_ThAr.lis
│   │   .
│   │   .
│   │   .
│   └── ThNe
│       ├── HR-R
│       │   └── HR-R_ThNe.lis
│       .
│       .
│       .
├── MasterBPM
│   └── master_bpm.fits
├── MasterBias
│   └── master_bias.fits
├── MasterFiberFlat
│   ├── LCB
│   │   └── HR-R
│   │       └── master_fiberflat.fits
│   └── MOS
│       └── LR-R
│           └── master_fiberflat.fits
├── MasterSensitivity
│   ├── LCB
│   │   └── HR-R
│   │       └── master_sensitivity.fits
│   └── MOS
│       └── LR-R
│           └── master_sensitivity.fits
├── MasterSlitFlat
│
├── MasterTwilightFlat
│   └── LCB
│       └── HR-R
│           └── master_twilightflat.fits
├── ModelMap
│   ├── LCB
│   │   └── HR-R
│   │       └── master_model.json
│   └── MOS
│       └── LR-R
│           └── master_model.json
├── TraceMap
│   ├── LCB
│   │   └── HR-R
│   │       └── master_traces.json
│   └── MOS
│       └── LR-R
│           └── master_traces.json
└── WavelengthCalibration
    ├── LCB
    │   └── HR-R
    │       └── master_wlcalib.json
    └── MOS
        └── LR-R
            └── master_wlcalib.json
```

Furthermore, the user's `MEGARA/` directory can contain data for your targets under different directories (in this example our targets are the M15 and M71 globular clusters). **Your raw data should always be included in a**

subdirectory named data/ within each working target directory (M15, M71, etc.). The different observation-result files (**.yaml*) used during the data reduction process should be also located within each target directory as they will be different for each target. In this example, the observation-result files in YAML format are named with a first number related in which they are run.

```
(megara) bash-3.2$ tree M15 M71 -L 2
M15
├── 0_bias.yaml
├── 1_tracemap.yaml
├── 2_modelmap.yaml
├── 3_wavecalib.yaml
├── 4_fiberflat.yaml
├── 5_twilight.yaml
├── 6_Lcbadquisition.yaml
├── 7_Standardstar.yaml
├── 8_reduce_LCB.yaml
└── data
    ├── 0001251794-20170626-MEGARA-MegaraLCBImage.fits
    ├── 0001251795-20170626-MEGARA-MegaraLCBImage.fits
    ├── 0001251796-20170626-MEGARA-MegaraLCBImage.fits
    ├── 0001286973-20170724-MEGARA-MegaraLcbImage.fits
    ├── 0001286974-20170724-MEGARA-MegaraLcbImage.fits
    ├── 0001286975-20170724-MEGARA-MegaraLcbImage.fits
    ├── ……
M71
├── 0_bias.yaml
├── 1_tracemap.yaml
├── 2_modelmap.yaml
├── 3_wavecalib.yaml
├── 4_fiberflat.yaml
├── 5_twilight.yaml
├── 6_Lcbadquisition.yaml
├── 7_Standardstar.yaml
├── 8_reduce_MOS.yaml
└── data
    ├── 0001287845-20170730-MEGARA-MegaraLcbImage.fits
    ├── 0001287846-20170730-MEGARA-MegaraLcbImage.fits
    ├── 0001287847-20170730-MEGARA-MegaraLcbImage.fits
    ├── 0001288184-20170731-MEGARA-MegaraMosImage.fits
    ├── 0001288185-20170731-MEGARA-MegaraMosImage.fits
    ├── ……
```

5.3 Running a recipe

The MEGARA DRP is run through a command line interface provided by **numina**.

The run mode of numina requires:

- An observation-result file in YAML format.
- A requirements file in YAML format (*control.yaml*).
- The raw images obtained as part of the user's observing block.
- The calibrations required by the recipe.

The observation-result file and the requirements file are created by the user. This is an example of the observation result file to compute the fibers traces:

```
(megara3) bash-3.2$ more 1_tracemap.yaml

id: 1_HR-R
```

```
mode: MegaraTraceMap
instrument: MEGARA
frames:
 - 0001312246-20170831-MEGARA-MegaraSuccess.fits
 - 0001312247-20170831-MEGARA-MegaraSuccess.fits
 - 0001312248-20170831-MEGARA-MegaraSuccess.fits
```

The "*id:"* is an identifier of the observing block. The DRP will create two directories with the products of the recipe (`/obsid_work` and `/obsid_results`) using the "*id"* identifier as a prefix to identify the corresponding processing block. The "*mode:"* is the name of the instrument observing mode as returned by `numina show-modes`. In "*frames:"* a list of the names of the images obtained as part of the observation should be included. Using the same YAML file the user can process sequentially different sets of files with the same recipe, the "*enabled:"* parameter can be set to *True* (or *False)* to process (or not) a specific block of files. Note that the used can add comments to these YAML files by adding lines preceded with a hash sign (#).

```
(megara) bash-3.2$ more 1_tracemap.yaml

id: 1_HR-R
mode: MegaraTraceMap
instrument: MEGARA
frames:
 - 0001312246-20170831-MEGARA-MegaraSuccess.fits
 - 0001312247-20170831-MEGARA-MegaraSuccess.fits
 - 0001312248-20170831-MEGARA-MegaraSuccess.fits
enabled: True
---
id: 1_HR-R_d29jun
mode: MegaraTraceMap
instrument: MEGARA
frames:
 - 0001252371-20170629-MEGARA-MegaraFiberFlatImage.fits
 - 0001252372-20170629-MEGARA-MegaraFiberFlatImage.fits
 - 0001252373-20170629-MEGARA-MegaraFiberFlatImage.fits
enabled: True
```

In the directory of our target M15 for example,

```
(megara) bash-3.2$ pwd
/Users/acm/Desktop/data_reduction/MEGARA/M15

(megara) bash-3.2$ ls

0_bias.yaml       2_modelmap.yaml    4_fiberflat.yaml   6_Lcbadquisition.yaml 8_reduce_LCB.yaml
1_tracemap.yaml   3_wavecalib.yaml   5_twilight.yaml    7_Standardstar.yaml   data
```

we run the recipe `MegaraTraceMap` using the observing-result file *1_tracemap.yaml* and the requirements file *control.yaml* in the following way:

```
(megara) bash-3.2$ numina run 1_tracemap.yaml -r ../control.yaml
```

Other useful numina commands include:

```
(megara) bash-3.2$ numina show-modes

(megara) bash-3.2$ numina show-recipes
```

5.4 Data reduction process

In the following sections the different steps to produce the target wavelength and flux calibrated row-stacked spectra (RSS) are detailed.

5.4.1 Bias image

Before the Analog-to-Digital conversion is performed a pedestal (electronic) level is added to all images obtained with the MEGARA CCD. This is a standard procedure in CCD imaging and spectroscopy applications for Astronomy and is intended to minimize the ADC errors produced when very low analog values are converted to DUs. To calibrate this pedestal level of the detectors, bias images are taking with null integration time. We note the user that in the case of the MEGARA CCD (a 4k x 4k pixels CCD231-84 E2V chip), since the detector is always read using two diagonally-opposed amplifiers (to speed up the reading process while minimizing electronic cross-talk), the bias is slightly different in the upper and bottom halves of the image.

This recipe processes a set of bias images obtained in Bias Image instrument mode. Images are corrected from overscan and trimmed to the physical size of the detector. Then, they are corrected from Bad-pixels Mask, if the BPM is available and finally, images are stacked using the median.

This is an example for the *0_bias.yaml*:

```
(megara) bash-3.2$ more 0_bias.yaml
id: 0_bias
mode: MegaraBiasImage
instrument: MEGARA
frames:
 - 0001310880-20170827-MEGARA-MegaraBiasImage.fits
 - 0001310881-20170827-MEGARA-MegaraBiasImage.fits
 - 0001310882-20170827-MEGARA-MegaraBiasImage.fits
 - 0001310883-20170827-MEGARA-MegaraBiasImage.fits
 - 0001310884-20170827-MEGARA-MegaraBiasImage.fits
 - 0001310885-20170827-MEGARA-MegaraBiasImage.fits
 - 0001310886-20170827-MEGARA-MegaraBiasImage.fits
 - 0001310887-20170827-MEGARA-MegaraBiasImage.fits
 - 0001310888-20170827-MEGARA-MegaraBiasImage.fits
```

The recipe is run as follows,

```
(megara) bash-3.2$ numina run 0_bias.yaml -r ../control.yaml
```

and the products are stored in the directory `obsid0_bias_results/`, including the *master_bias.fits* file (see ***Figure 4***). The user needs to copy this file to the calibration tree at `ca3558e3-e50d-4bbc-86bd-da50a0998a48/MasterBias/`.

```
(megara) bash-3.2$ tree obsid0_bias_work/ obsid0_bias_results/
obsid0_bias_work/
├── 0001310880-20170827-MEGARA-MegaraBiasImage.fits
├── 0001310881-20170827-MEGARA-MegaraBiasImage.fits
├── 0001310882-20170827-MEGARA-MegaraBiasImage.fits
├── 0001310883-20170827-MEGARA-MegaraBiasImage.fits
├── 0001310884-20170827-MEGARA-MegaraBiasImage.fits
├── 0001310885-20170827-MEGARA-MegaraBiasImage.fits
├── 0001310886-20170827-MEGARA-MegaraBiasImage.fits
├── 0001310887-20170827-MEGARA-MegaraBiasImage.fits
├── 0001310888-20170827-MEGARA-MegaraBiasImage.fits
├── index.pkl
└── master_bpm.fits
```

```
obsid0_bias_results/
├── master_bias.fits
├── processing.log
├── result.yaml
└── task.yaml
```

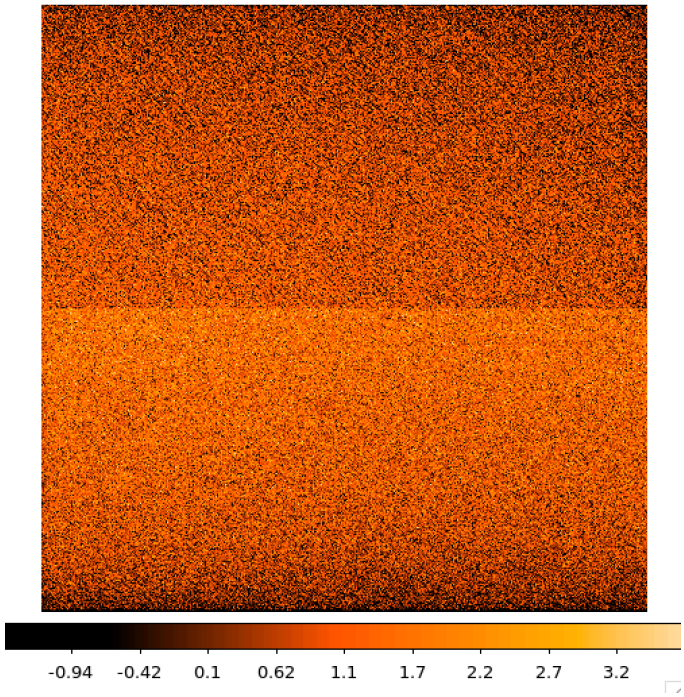


Figure 4: *Example of a MEGARA master bias as created by the MegaraBiasImage recipe.*

5.4.2 Dark image

The potential wells in CCD detectors spontaneously generate electron-ion pairs at a rate that is a function of temperature. For very long exposures this translates into a current that is associated with no light source and that is commonly referred to as dark current. Different tests during AIV activities have shown MEGARA detector´s dark current has very low values < 2 e^-/h/pixel, therefore in our data reduction dark images are neither needed nor used.

5.4.3 Bad-pixels Mask

Although science-grade CCD detectors show very few bad pixels / bad columns there will be a number of pixels (among the ~17 Million pixels in the MEGARA CCD) whose response could not be corrected by means of using calibration images such as dark frames or flat-field images. These pixels, commonly called either dead or hot pixels, should be identified and masked so their expected signal could be derived using dithered images or, alternatively, locally interpolated. The user is provided with a bad-pixels mask (*master_bpm.fits*) located at `ca3558e3-e50d-4bbc-86bd-da50a0998a48/MasterBPM/` that was generated as part of the AIV activities by processing a set of defocused continuum flat images. Currently, MEGARA presents only one (partial) bad column of 120 pixels in length.

5.4.4 Slit Flat correction

In the case of fiber-fed spectrographs the correction for the detector pixel-to-pixel variation of the sensibility is usually carried out using data from laboratory, where the change in efficiency of the detector at different

wavelengths is computed and then used to correct for this effect for each specific instrument configuration (VPH setup in the case of MEGARA).

The quality of present-day CCDs leads to a rather small impact of these pixel-to-pixel variations in sensitivity on either the flux calibration and the cosmetics of the scientific images, especially considering that not one but a number of pixels along the spatial direction are extracted for each fiber and at each wavelength. In the case of MEGARA, the pseudo-slit has been offset from its optical focus position to ensure that the gaps between fibers are also illuminated when a continuum (halogen) lamp at the ICM is used. The results of the analysis of the pixel-to-pixel variations in sensitivity show that this correction is actually not needed although this recipe is implemented in the MEGARA DRP.

5.4.5 Tracing fibers

5.4.5.1 Trace map

The next processing step combine a series of fiber-flats to generate a master "trace map". The fiber-flats are obtained by illuminating the instrument focal plane with a continuum (halogen) lamp that is part of the GTC Instrument Calibration Module (ICM).

This step produces the tracing information required to extract the flux of the fibers. The result is stored in a file named *master_traces.json*.

An example of the observation result file *1_tracemap.yaml* to trace the fibers is the following:

```
(megara) bash-3.2$ more 1_tracemap.yaml
id: 1_HR-R
mode: MegaraTraceMap
instrument: MEGARA
frames:
 - 0001312246-20170831-MEGARA-MegaraSuccess.fits
 - 0001312247-20170831-MEGARA-MegaraSuccess.fits
 - 0001312248-20170831-MEGARA-MegaraSuccess.fits
```

Then the recipe is run by doing:

```
(megara) bash-3.2$ numina run 1_tracemap.yaml -r ../control.yaml
```

Images listed in the observation-result file are trimmed and corrected from overscan, bad-pixel mask (if *master_bpm* is present), bias and dark current (if *master_dark* is present). Images thus corrected are then median stacked. The result of the combination is saved as an intermediate result that is named '*reduced_image.fits'*. This combined image is also returned in the field *reduced_image* of the recipe result and will be used for doing some quality control on the tracing of the fibers.

The fibers are then grouped in packs of different numbers of fibers. To match the traces in the image with the corresponding fibers, the DRP uses the information provided by the instrument configuration to know how fibers are packed and where the different groups of fibers appear in the detector. Using the column reference 2000, peaks are detected (using an average of 7 columns) and matched to the layout of fibers. Fibers without a matching peak are counted and their ids stored in the final *master_traces.json* file. Once the peaks in the reference column are found, each one is traced until the border of the image is reached. The trace may be lost before reaching the border. In all cases, the beginning and the end of the trace are stored.

The Y position of the trace is fitted to a polynomial of degree polynomial_degree set to 5 by default. The coefficients of the polynomial are stored in the final *master_traces.json* file.

```
(megara) bash-3.2$ tree obsid1_HR-R_work/ obsid1_HR-R_results/ -L 2
obsid1_HR-R_work/
├── 0001312246-20170831-MEGARA-MegaraSuccess.fits
├── 0001312247-20170831-MEGARA-MegaraSuccess.fits
├── 0001312248-20170831-MEGARA-MegaraSuccess.fits
├── ds9.reg
├── ds9_raw.reg
├── index.pkl
├── master_bias.fits
├── master_bpm.fits
└── reduced_image.fits
obsid1_HR-R_results/
├── master_traces.json
├── processing.log
├── reduced_image.fits
├── reduced_rss.fits
├── result.yaml
└── task.yaml
```

The position of the fibers traces at the detector are shifted depending on the ambient temperature. It is recommended to have continuum halogen exposures near in time to the observation of the scientific target. If this is not the case, the traces can be shifted easily when processing the target (see section 5.4.5.2).

The traces generated by this task can be visualized both on the raw or the processed images and can be also shifted to consider possible offsets between these traces and the position in the fibers in other images (twilight flats, standard star or scientific target observations, etc.). The visualization of the traces and an underlying reduced image can be done by executing:

```
(megara)bash-3.2$ megaradrp-overplot_traces reduced_image.fits master_traces.json
```

or

```
(megara)bash-3.2$ megaradrp-overplot_traces --rawimage 0001312246-20170831-MEGARA-MegaraSuccess.fits
master_traces.json
```

respectively for the reduced and raw images. Another way to check the tracing is by overplotting the ds9 region files created by the DRP for the traces on top of this *reduced_image* by doing (syntax might vary):

```
(megara) bash-3.2$ ds9 obsid1_HR-R_results/reduced_image.fits -regions load obsid1_HR-R_work/ds9.reg
```

The same syntax can be used to check the offset between these traces and the position of the fibers in other images (arc-lamp, twilight, standard star and object images).

Finally, the user needs to copy this *master_traces.json* to the corresponding place at the calibration tree.

```
(megara)bash-3.2$ cd obsid1_HR-R_results/

(megara)bash-3.2$ cp master_traces.json ../../ca3558e3-e50d-4bbc-86bd-da50a0998a48/TraceMap/LCB/HR-R
```

5.4.5.2 Model map

This recipe processes a set of continuum flat images obtained in *Trace Map* or *Fiber Flat* modes and returns the fiber profile information required to perform **advanced** fiber extraction in other recipes.

The set of files listed in the observation-result file *2_modelmap.yaml* is the same one used for the Trace Map.

```
(megara) bash-3.2$ more 2_modelmap.yaml
id: 2_HR-R
mode: MegaraModelMap
instrument: MEGARA
frames:
 - 0001312246-20170831-MEGARA-MegaraSuccess.fits
 - 0001312247-20170831-MEGARA-MegaraSuccess.fits
 - 0001312248-20170831-MEGARA-MegaraSuccess.fits
```

Then the recipe is run by doing:

```
(megara) bash-3.2$ numina run 2_modelmap.yaml -r ../control.yaml
```

This processing step might take several minutes (from 10-40 min.) depending on the hardware used. When a model map is used the running times of the subsequent processing steps also increase by 2-5 minutes.

The images are processed as in the Trace Map recipe. In this case, the approximate central position of the fibers is obtained from the previously computed *master_traces.json*. Then, for every 100 columns of the reduced image, a vertical cut in the image is fitted to a sum of fiber profiles, being the profile a gaussian convolved with a square. After the columns are fitted, the profiles (central position and sigma) are interpolated to all columns using splines (see ***Figure 5***). The coefficients of the resulting splines are stored in the final *master_model.json* file.

The recipe also returns the RSS obtained by applying this advanced extraction to *reduced_image*. As an intermediate result, the recipe produces DS9 region files with the position of the center of the profiles, that can be used with raw and reduced images (see ***Figure*** 6).

```
(megara) bash-3.2$ tree obsid2_HR-R_work/ obsid2_HR-R_results/ -L 2
obsid2_HR-R_work/
├── 0001312246-20170831-MEGARA-MegaraSuccess.fits
├── 0001312247-20170831-MEGARA-MegaraSuccess.fits
├── 0001312248-20170831-MEGARA-MegaraSuccess.fits
├── ds9.reg
├── ds9_raw.reg
├── fib_100_mean.png
├── fib_100_std.png
├── fib_101_mean.png
├── fib_101_std.png
├── fib_102_mean.png
├── fib_102_std.png
├── …
├── index.pkl
├── master_bias.fits
├── master_bpm.fits
└── reduced_image.fits
obsid2_HR-R_results/
├── master_model.json
├── processing.log
├── reduced_image.fits
├── reduced_rss.fits
├── result.yaml
└── task.yaml
```

The user needs to copy this *master_model.json* to the corresponding place at the calibration tree.

```
(megara)bash-3.2$ cd obsid2_HR-R_results/

(megara)bash-3.2$ cp master_model.json ../../ca3558e3-e50d-4bbc-86bd-da50a0998a48/ModelMap/LCB/HR-R
```

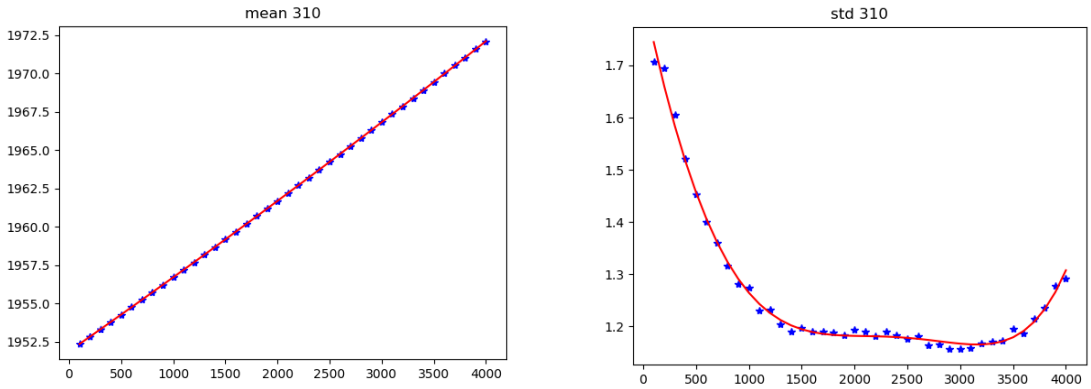


Figure 5: *Mean position (left) and sigma (right) in pixels for fiber #310 along the spectral axis shown as blue points. The red line shows the spline fit. Plots for all the fibers are stored in the* `obsid_work/` *directory.*

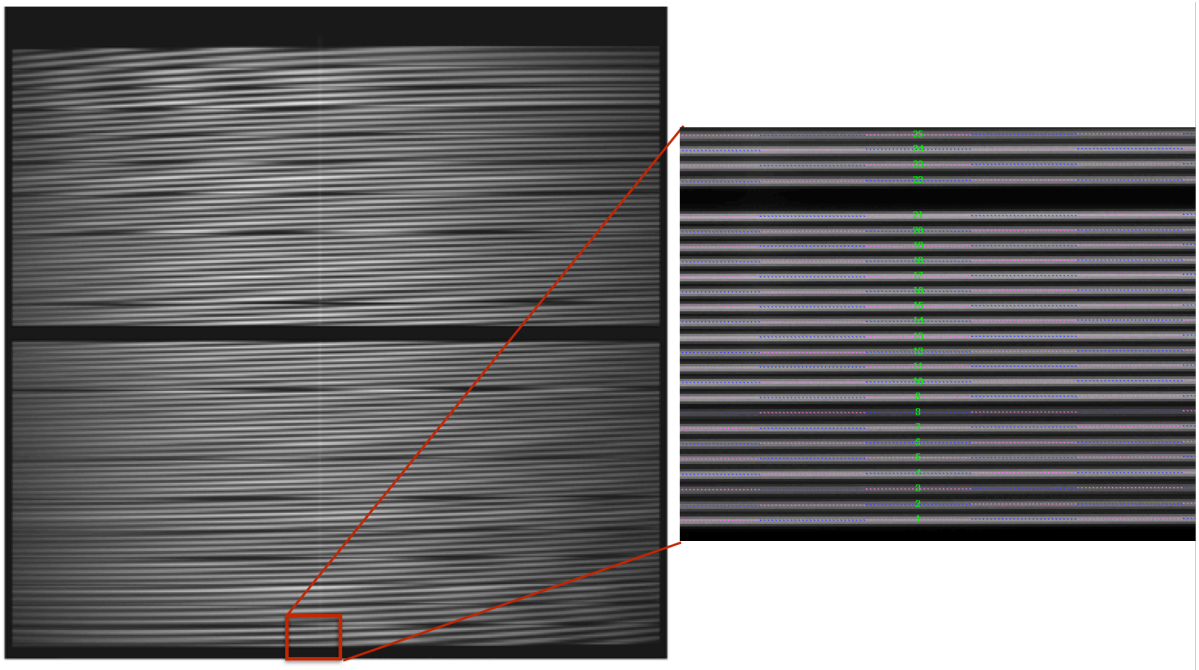


Figure 6: *MEGARA LCB HR-R continuum halogen exposure (left) and a region of the raw image (right) with the ds9_raw.reg tracing the fibers' path shown on top.*

5.4.6 Wavelength Calibration

In this processing step the wavelength solution for each fiber is created using recipe *MegaraArcCalibration*. To create the dispersion solution the recipe needs raw arc-lamp[9] frames as input (see ***Figure 7***).

[9] Note that although the term used is "arc-lamps" these are ThAr and ThNe Hollow-Cathode Lamps (HCL).

The user needs to check if the traces already computed in the previous step are appropriate to do the extraction in the arc-lamp exposures. If the continuum halogen used to generate the traces and the arc-lamp images were obtained near in time there no offset should be applied to the traces[10]. The user can check this and evaluate the actual offset by plotting the *ds9_raw.reg* regions file on top of the arc-lamp raw image using DS9. If the traces (regions in *ds9_raw.reg*) are above the fiber as seen in the raw image, then the offset is a negative number and it is measured in pixels, while if the traces are below then the offset is a positive number. This offset is given in the *"requirements"* section in the observation-result file using the *"extraction_offset"* parameter.

In this case, the observation-result file is called *3_wavecalib.yaml*. In the example below, three frames for arc lamp exposures are included and the offset for the extraction is set to 0 pixels:

```
(megara) bash-3.2$ more 3_wavecalib.yaml
id: 3_HR-R
mode: MegaraArcCalibration
instrument: MEGARA
frames:
 - 0001312249-20170831-MEGARA-MegaraSuccess.fits
 - 0001312250-20170831-MEGARA-MegaraSuccess.fits
 - 0001312251-20170831-MEGARA-MegaraSuccess.fits
requirements:
 extraction_offset: [0.0]
 store_pdf_with_refined_fits: 1
```

Then the recipe is run by doing:

```
(megara) bash-3.2$ numina run 3_wavecalib.yaml -r ../control.yaml
```

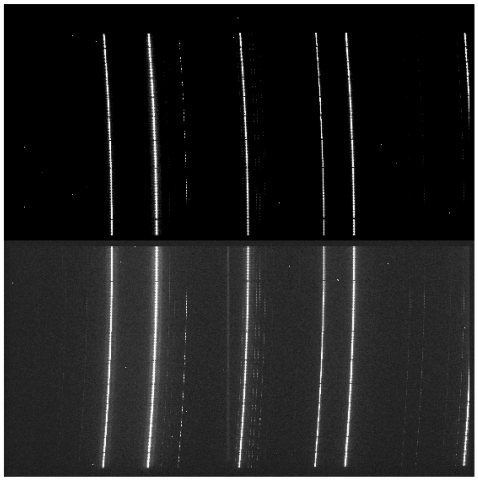


Figure 7: *MEGARA LCB ThNe arc-lamp exposure obtained with the HR-R VPH.*

Images provided in *3_wavecalib.yaml* are trimmed and corrected from overscan, bad-pixel mask (if *master_bpm* is present), bias and dark current (if *master_dark* is present). The corrected images are then stacked using a median. The result of the combination of these images is saved as an intermediate result, named *'reduced_image.fits'*.

[10] By these being taken near in time the user ensures that the operating temperature remains constant and no offset is present. The offsets are estimated to be about 1 pixel per degree of change in the operating temperature. No offset in the spectral direction has been reported.

The apertures in the 2D image are extracted, using the information in *master_traces.json* (or in the *model_map.json* if this file is present at the calibration tree) and the *"extraction_offset"* parameter set in the *3_wavecalib.yaml*. The result of the extraction is saved as an intermediate result named *'reduced_rss.fits'*. The requirement file *control.yaml* has useful information for the wavelength calibration. For each fiber in the reduced RSS, the peaks are detected and sorted by peak intensity. Then, a total of *nlines* as listed in the *control.yaml* file are used to select the brightest peaks. If it is a list, then the peaks are divided, by their position, in as many groups as elements in the list and *nlines[0]* peaks are selected in the first group, *nlines[1]* peaks in the second, etc. The selected peaks are then matched against the catalog of lines located in the calibration tree at `ca3558e3-e50d-4bbc-86bd-da50a0998a48/LinesCatalog/`. The wavelengths of the matched features are fitted to a polynomial of degree equal to *polynomial_degree*. The matched lines, the quality of the match and other relevant information such as the coefficients of the polynomial are stored in the final *master_wlcalib.json* for each fiber.

Finally, the recipe returns different products. At the `obsid_work/` directory the files *wavecal_iter1.pdf* (for the initial wavelength calibration) and *wavecal_iter2.pdf* (for the final iteration) contain a graphical representation for the wavelength calibration for each fiber. For example, in *wavecal_iter2.pdf* the total number of lines used for the refined wavelength calibration and the root mean square for each fit is plotted depending on the fiber number. In the same PDF file, the linear approximation for CRVAL1 and CDELT1 is plotted and also a graph for each coefficient (typically of 5th degree) of the polynomial fit used for the refined wavelength calibration is shown (see ***Figure 8***).

Should the user set the *store_pdf_with_refined_fits* parameter to *"store_pdf_with_refined_fits: 1"* at the *3_wavecalib.yaml*, the recipe will create the subdirectory `obsid3_HR-R_work/refined_wavecal/` where a collection of PDF files (one for each fiber) is created with graphical information about the refined wavelength calibration (see ***Figure 9***).

```
(megara) bash-3.2$ tree obsid3_HR-R_work/ obsid3_HR-R_results/ -L 2
obsid3_HR-R_work/
├── 0001312249-20170831-MEGARA-MegaraSuccess.fits
├── 0001312250-20170831-MEGARA-MegaraSuccess.fits
├── 0001312251-20170831-MEGARA-MegaraSuccess.fits
├── index.pkl
├── initial_master_wlcalib.json
├── master_bias.fits
├── master_bpm.fits
├── reduced_image.fits
├── reduced_rss.fits
├── refined_wavecal
│   ├── 001.pdf
│   ├── 002.pdf
│   ├── 003.pdf
│   ├── 004.pdf
│   ├── 005.pdf
│   …
├── wavecal_iter1.pdf
└── wavecal_iter2.pdf
obsid3_HR-R_results/
├── fwhm_image.fits
├── master_wlcalib.json
├── processing.log
├── reduced_image.fits
├── reduced_rss.fits
├── result.yaml
└── task.yaml
```

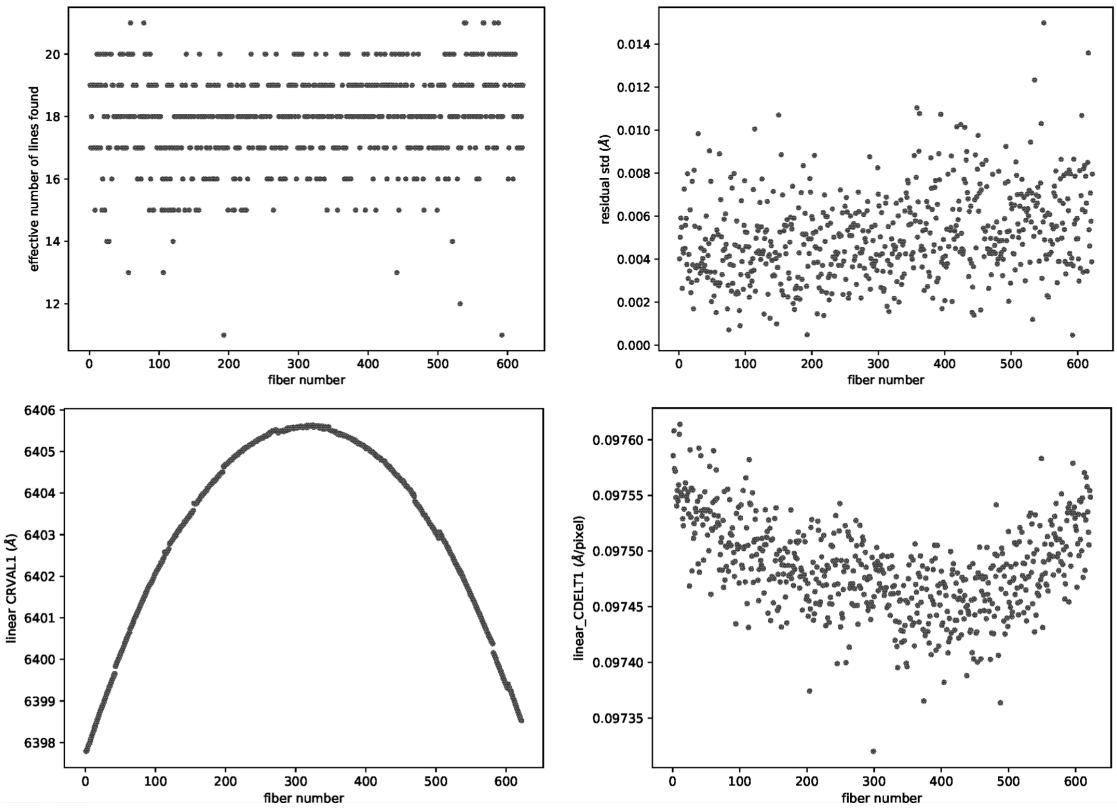


Figure 8: *Some of the plots included in wavecalib_iter2.pdf file generated with the MegaraArcCalibration recipe.*

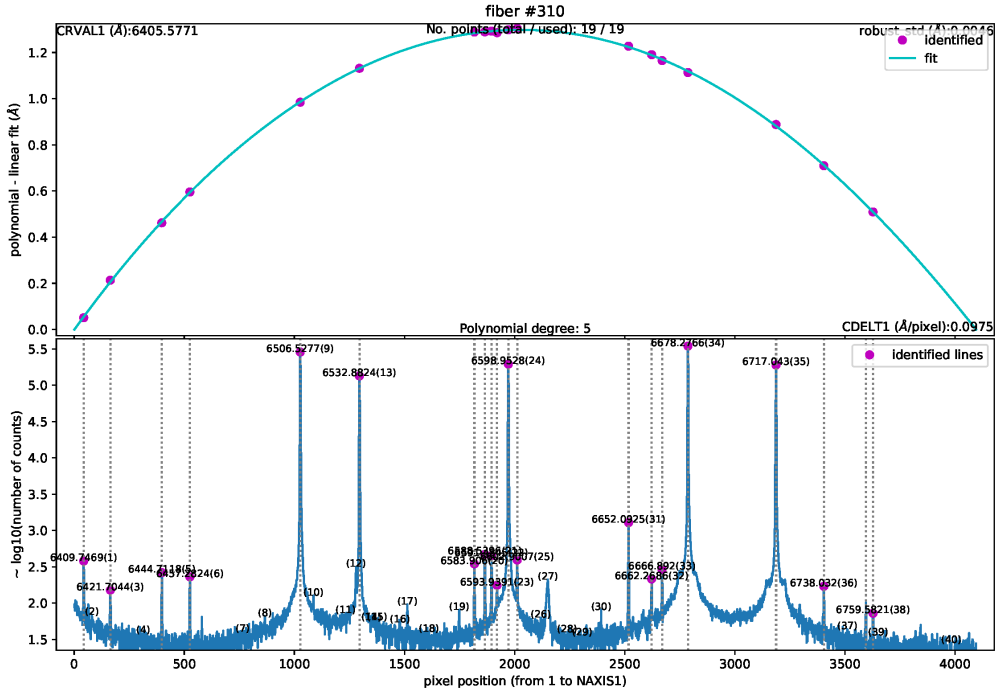


Figure 9: *Example of the refined wavelength calibration result for fiber #310. This kind of file (310.pdf at refined_wavecalib/ in this case) is generated when the parameter "store_pdf_with_refined_fits" is set to 1. This requirement should be set to 0 for a fast execution of this recipe.*

The user needs to copy the *master_wlcalib.json* at the `obsid_result/` directory to the corresponding place at the calibration tree:

```
(megara)bash-3.2$ cd obsid3_HR-R_results/

(megara)bash-3.2$ cp master_wlcalib.json ../../ca3558e3-e50d-4bbc-86bd-da50a0998a48/WavelengthCalibration/LCB/HR-R
```

5.4.7 Flat-field correction

Each optical fiber in MEGARA behaves like a different optical system, and therefore, its optical transmission is different and individual, with different wavelength dependence.

The recipe *MegaraFiberFlatImage* computes the *master_fiberflat.fits* to correct for the global variations in transmission in between fibers and as a function of wavelength in MEGARA. A fiber-flat image should be used to perform this correction. These images are obtained by means of illuminating the instrument focal plane with a flat spectral source (typically a halogen lamp) that is installed as part of the GTC Instrument Calibration Module (ICM).

In this case, we called the observation result file *4_fiberflat.yaml*, where a total of three continuum halogen exposures are included. If the inputs frames are the same used to trace the fiber spectra on the detector for the same specific spectral setup, the *"extraction_offset"* parameter should be set to 0 pixels. If that is note the case the offset should be evaluated and computed as detailed in last section 5.4.6.

```
(megara) bash-3.2$ more 4_fiberflat.yaml
id: 4_HR-R
mode: MegaraFiberFlatImage
instrument: MEGARA
frames:
 - 0001312246-20170831-MEGARA-MegaraSuccess.fits
 - 0001312247-20170831-MEGARA-MegaraSuccess.fits
 - 0001312248-20170831-MEGARA-MegaraSuccess.fits
requirements:
 extraction_offset: [0.0]
```

Then the recipe is run by doing:

```
(megara) bash-3.2$ numina run 4_fiberflat.yaml -r ../control.yaml
```

All images listed in the observation-result file are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is present), bias and dark current (if *master_dark* is present) and corrected from pixel-to-pixel flat if *master_slitflat* is provided. The corrected images are then stacked using a median. The result of the combination is saved as an intermediate result, named *'reduced_image.fits'*.

The apertures in the 2D image are extracted, using the information in *master_traces.json* (or in the *model_map.json* if this file is present at the calibration tree) and the *"extraction_offset"* parameter set in the *4_fiberflat.yaml,* and then it is resampled according to the wavelength calibration in *master_wlcalib.json*. The resulting RSS is saved as an intermediate result named *'reduced_rss.fits'*. To normalize the *master_fiberflat*, each fiber is divided by the best-fitting spline to the average of all valid fibers (see ***Figure 10***). The RSS image *master_fiberflat.fits* is returned as a recipe result (see ***Figure 11***).

```
(megara) bash-3.2$ tree obsid4_HR-R_work/ obsid4_HR-R_results/ -L 2
obsid4_HR-R_work/
├── 0001312246-20170831-MEGARA-MegaraSuccess.fits
├── 0001312247-20170831-MEGARA-MegaraSuccess.fits
├── 0001312248-20170831-MEGARA-MegaraSuccess.fits
├── collapse.txt
├── collapsed_smooth.png
├── index.pkl
├── mask_noinfo.txt
├── master_bias.fits
├── master_bpm.fits
├── reduced_image.fits
└── reduced_rss.fits
obsid4_HR-R_results/
├── master_fiberflat.fits
├── processing.log
├── reduced_image.fits
├── reduced_rss.fits
├── result.yaml
└── task.yaml
```

The user needs to copy the *master_fiberflat.json* at the `obsid_result/` directory to the corresponding place at the calibration tree:

```
(megara)bash-3.2$ cd obsid4_HR-R_results/

(megara)bash-3.2$ cp master_fiberflat.json ../../ca3558e3-e50d-4bbc-86bd-da50a0998a48/MasterFiberFlat/LCB/HR-R
```

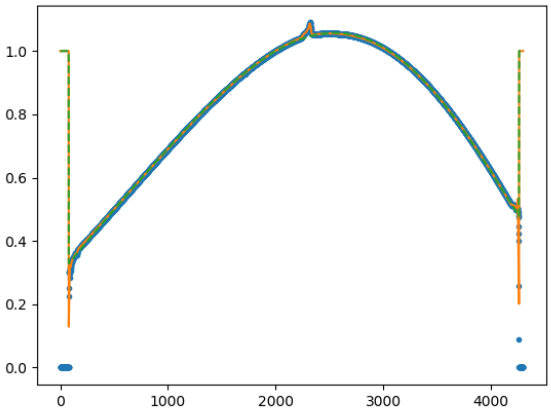


Figure 10: *Example of the collapsed_smooth.png file generated as part of the MegaraFiberFlat recipe, which is located at the* `obsid_work/` *directory. The green line is a spline fit to the average of all valid fibers, which is then used to normalize the extracted spectral in order to generate the normalized master_fiberflat image.*

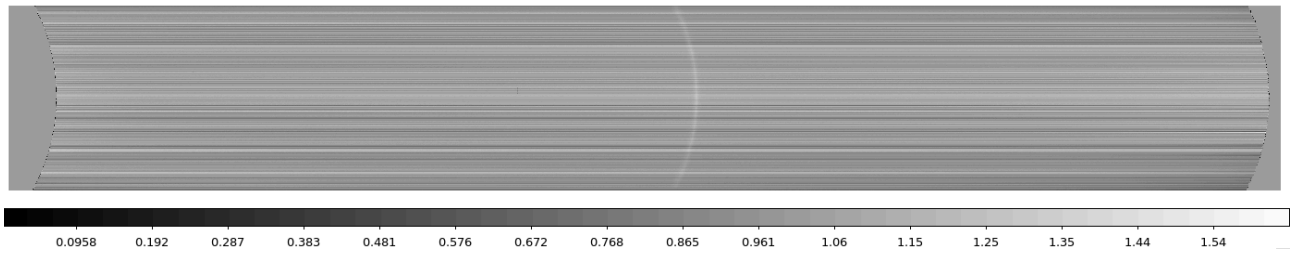


Figure 11: *Example of the master_fiberflat.fits file generated for MEGARA LCB HR-R mode.*

5.4.8 Illumination correction

Blank twilight-sky exposures are to be used to calibrate the global change in response introduced by the fiber flat. This is called the illumination correction and it is due to the fact that the GTC ICM does not produce a perfectly uniform illumination of the field and that the fraction and shape of the pupil that is seen by the MEGARA fibers during the observation of a specific target does not coincide with that seen during the acquisition of the fiber-flat images with the ICM.

The twilight sky exposure can safely assume to homogeneously illuminate the entire MEGARA field of view (3.5 arcmin x 3.5 arcmin for MOS mode and 12.5 x 11.3 sq. arcsec for LCB mode). However, since the telescope pupil is not circular and the alignment of the image of the pupil on top fibers by the microlenses is not identical for all fibers, in order to do this correction properly, the Rotator Angle of the FC-F rotator (ROTANG keyword in the raw image) and the Elevation of the telescope (ELEVAT keyword), and ideally also the temperature, should have the same values as the ones for the scientific observation. Furthermore, in case of MOS observing mode, the twilight sky exposures should be done with the robotic positioners placed at the same positions as for the targets' configuration.

The recipe *MegaraTwilightFlatImage* process a set of continuum blank twilight sky images and returns the master twilight flat product. In this case, we named observation result file as *5_twilight.yaml*, where three frames for continuum blank twilight sky exposures being listed in the file. The *"extraction_offset"* parameter can be computed as detailed in section 5.4.6 (see ***Figure 12***).

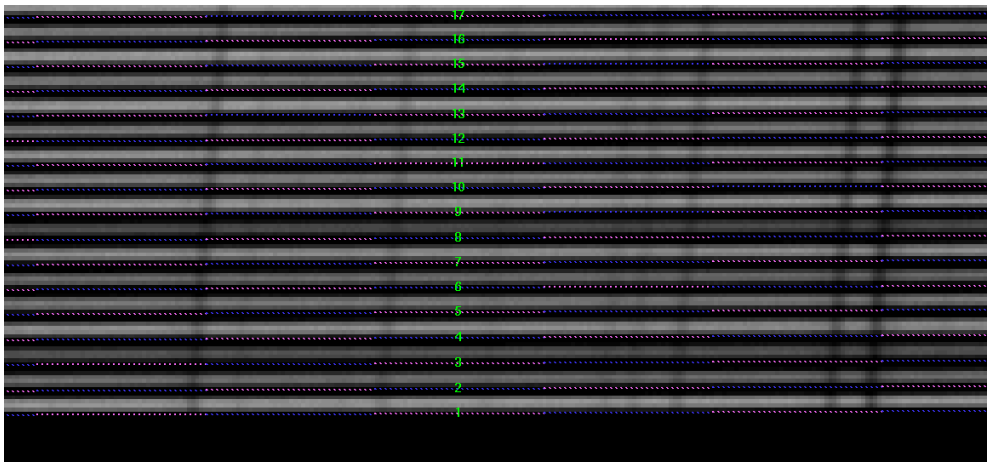


Figure 12: *Example of a region in the raw blank twilight sky image (LCB, HR-R) with the computed traces (ds9_raw.reg file) on top. In this case a "extraction _offset" of +2.5 pixels was needed.*

```
(megara) bash-3.2$ more 5_twilight.yaml
id: 5_HR-R
mode: MegaraTwilightFlatImage
instrument: MEGARA
frames:
 - 0001251794-20170626-MEGARA-MegaraLCBImage.fits
 - 0001251795-20170626-MEGARA-MegaraLCBImage.fits
 - 0001251796-20170626-MEGARA-MegaraLCBImage.fits
requirements:
  extraction_offset: [+2.5]
  normalize_region: [1550,1700]
  continuum_region: [1750,1900]
```

Then the recipe is run by doing:

```
(megara) bash-3.2$ numina run 5_twilight.yaml -r ../control.yaml
```

Images provided in the observation-result file are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is present), bias and dark current (if *master_dark* is present) and corrected from pixel-to-pixel flat if *master_slitflat* is provided. The corrected images are then stacked using a median. The result of the combination is saved as an intermediate result, named *'reduced_image.fits'*.

The apertures in the 2D image are extracted, using the information in *master_traces.json* (or in the *model_map.json* if this file is present at the calibration tree) and the *"extraction_offset"* parameter set in the *5_twilight.yaml*, and then it is resampled according to the wavelength calibration in *master_wlcalib.json*. Then, the result is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named *'reduced_rss.fits'*. To normalize the *master_twilightflat* (see ***Figure 13***), each fiber is divided by the average of the column range given in "*normalize_region"* parameter in *5_twilight.yaml*. In those cases where the observation of an object includes a bright sky line, this "*normalize_region"* parameter can be used to obtain a twilight flat image from these science observations, especially if twilight frames of the same ROTANG, ELEVAT and temperature values are not available. In that case, the user can also make use of the parameter "*continuum_region*" to previously subtract the sky continuum under the bright sky line of interest. Note that these using are pixels in the x-axis of the *"reduced_image.fits"* image.

```
(megara) bash-3.2$ tree obsid5_HR-R_work/ obsid5_HR-R_results/ -L 2
obsid5_HR-R_work/
├── 0001251794-20170626-MEGARA-MegaraLCBImage.fits
├── 0001251795-20170626-MEGARA-MegaraLCBImage.fits
├── 0001251796-20170626-MEGARA-MegaraLCBImage.fits
├── index.pkl
├── master_bias.fits
├── master_bpm.fits
├── master_fiberflat.fits
├── reduced_image.fits
└── reduced_rss.fits
obsid5_HR-R_results/
├── master_twilightflat.fits
├── processing.log
├── reduced_image.fits
├── reduced_rss.fits
├── result.yaml
└── task.yaml
```

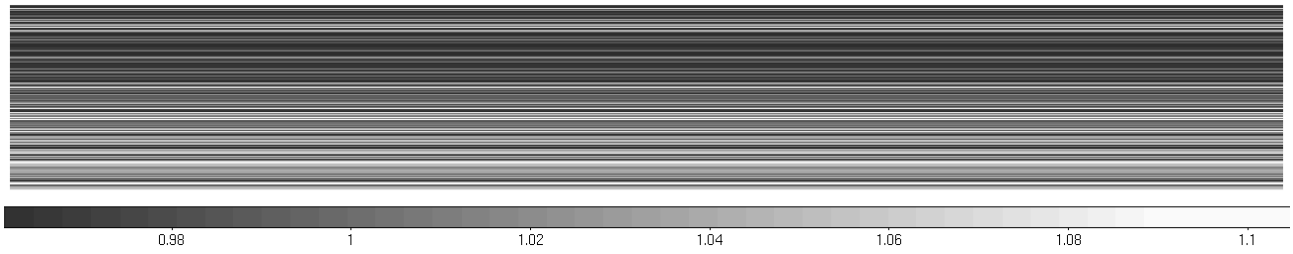


Figure 13: *Example of the master_twilightflat.fits file generated for MEGARA LCB HR-R mode.*

The user needs to copy the *master_twilightflat.fits* at the `obsid_result/` directory to the corresponding place at the calibration tree:

```
(megara)bash-3.2$ cd obsid5_HR-R_results/

(megara)bash-3.2$ cp master_twilightflat.fits ../../ca3558e3-e50d-4bbc-86bd-da50a0998a48/MasterTwilight/LCB/HR-R
```

5.4.9 Flux calibration

The flux calibration is performed by observing one or several spectrophotometric stars with the same instrument configuration that for the scientific observations. Depending on the number of standard stars observed and on the weather conditions (mainly transparency) two different types of calibration could be achieved:

- Absolute-flux calibration: The weather conditions during the night should be photometric and a number of spectrophotometric standard stars at different airmasses should be observed. This allows to fully correct from DUs per CCD pixel to energy surface density (typically in AB magnitudes, Jankys or erg s^{-1} cm^{-2} $Å^{-1}$) incident at the top of the atmosphere. If only one single standard star is observed (ideally at the airmass of the science object) this correction allows deriving the energy surface density hitting the telescope primary mirror exclusively, unless an atmospheric extinction curve for the observatory and that particular night is assumed (in which case the airmass could be different). In order to properly flux-calibrate scientific observations at all airmasses several stars should be observed during the night.
- Relative-flux calibration: If the weather conditions are not photometric this correction only allows normalizing the DUs per CCD pixel along the spectral direction so the conversion to incident energy at the top of the atmosphere is the same at all wavelengths. In order for this calibration to be valid one must assume that the effect of the atmosphere (including atmospheric cirrus and possibly thick clouds) on the wavelength dependence of this correction is that given by the adopted atmospheric extinction curve, even if the absolute flux level is not.

In the following, the different steps to do an absolute flux calibration are described. A photometric night and one spectrophotometric standard star observation with the same airmass as the scientific observation are assumed.

The entire flux of the spectrophotometric standard star needs to be recovered, so the LCB IFU bundle must be used. The recipe *MegaraLcbAcquisition* is used to process and extract the spectra in the standard star observation and determine the position of the centroid of the target in the LCB field of view, around which the total flux of the star will be later recovered.

In this case, the observation-result file for determining the star centroid is *6_lcbadquisition.yaml*, where three frames for spectrophotometric standard star exposures are here listed. The *"extraction_offset"* parameter can be computed as detailed in section 5.4.6.

```
(megara) bash-3.2$ more 6_Lcbadquisition.yaml
id: 6_HR-R
mode: MegaraLcbAcquisition
instrument: MEGARA
frames:
 - 0001286973-20170724-MEGARA-MegaraLcbImage.fits
 - 0001286974-20170724-MEGARA-MegaraLcbImage.fits
 - 0001286975-20170724-MEGARA-MegaraLcbImage.fits
requirements:
 extraction_offset: [+4.5]
```

Then the recipe is run by doing:

```
(megara) bash-3.2$ numina run 6_lcbadquisition.yaml -r ../control.yaml
```

Images provided in observation-result file are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is present), bias and dark current (if *master_dark* is present) and corrected from pixel-to-pixel flat if *master_slitflat* is provided. The corrected images are then stacked using a median. The result of the combination is saved as an intermediate result, named *'reduced_image.fits'*.

The apertures in the 2D image are extracted, using the information in *master_traces.json* (or in the *model_map.json* if this file is present at the calibration tree) and the *"extraction_offset"* parameter set in the *6_lcbadquisition.yaml,* and then it is resampled according to the wavelength calibration in *master_wlcalib.json*. Then it is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named *'reduced_rss.fits'*.

The sky is subtracted by combining the 56 fibers dedicated for this purpose in the LCB mode. The RSS with sky subtracted is saved in a file named *'final_rss.fits'* as a result of the recipe. Then, the centroids around both the center of the field and the brighest spaxel are computed using up the signal from the 3 rings of fibers (37 fibers in total) around these two spaxels. The offsets needed to center the object (considered to be either the centroid around the central spaxel or, more likely, around the brightest spaxel) in the center of the LCB field are then returned both in mm and arcsec. This information is saved in the *"processing.log"* file at the `obsid_result/` directory.

```
(megara) bash-3.2$ tree obsid6_HR-R_work/ obsid6_HR-R_results/ -L 2
obsid6_HR-R_work/
├── 0001286973-20170724-MEGARA-MegaraLcbImage.fits
├── 0001286974-20170724-MEGARA-MegaraLcbImage.fits
├── 0001286975-20170724-MEGARA-MegaraLcbImage.fits
├── index.pkl
├── master_bias.fits
├── master_bpm.fits
├── master_fiberflat.fits
├── master_twilightflat.fits
├── reduced_image.fits
└── reduced_rss.fits
obsid6_HR-R_results/
├── final_rss.fits
├── processing.log
├── reduced_image.fits
├── reduced_rss.fits
├── result.yaml
└── task.yaml


(megara) bash-3.2$ cd obsid6_HR-R_results/


(megara) bash-3.2$ more processing.log
2018-08-14 10:19:36,656 - numina.recipes.megara - INFO - end sky subtraction
2018-08-14 10:19:36,837 - numina.recipes.megara - DEBUG - LCB configuration is b7dcd9d1-0b60-4b43-b26e-d2c9868d5e20
2018-08-14 10:19:36,837 - numina.recipes.megara - DEBUG - unit is arcsec
2018-08-14 10:19:36,838 - numina.recipes.megara - INFO - maximum flux in spaxel 311 -- unknown
2018-08-14 10:19:36,842 - numina.recipes.megara - INFO - Using 37 nearest fibers
2018-08-14 10:19:36,842 - numina.recipes.megara - INFO - For point [0, 0] arcsec
2018-08-14 10:19:36,842 - numina.recipes.megara - INFO - For point [0.0, 0.0] mm
2018-08-14 10:19:36,843 - numina.recipes.megara - DEBUG - nearest fibers
2018-08-14 10:19:36,843 - numina.recipes.megara - DEBUG - [310, 313, 311, 312, 308, 309, 314, 315, 307, 316, 288, 289, 333, 334, 305,
306, 317, 318, 292, 293, 329, 330, 304, 319, 335, 336, 290, 291, 331, 332, 296, 297, 325, 326, 302, 303, 321]
2018-08-14 10:19:36,843 - numina.recipes.megara - INFO - centroid: [0.2920111992228447, 0.05845381069681873] arcsec
2018-08-14 10:19:36,844 - numina.recipes.megara - INFO - centroid: [0.260724285020397, 0.052190902407873864] mm
2018-08-14 10:19:36,845 - numina.recipes.megara - INFO - 2nd order moments, x2=0.345658, y2=0.357311, xy=-0.006625 arcsec^2
2018-08-14 10:19:36,845 - numina.recipes.megara - INFO - FWHM , x=1.384461, y=1.407606 arcsec
2018-08-14 10:19:36,845 - numina.recipes.megara - INFO - For point [0.465000003576279, 0.0] arcsec
2018-08-14 10:19:36,845 - numina.recipes.megara - INFO - For point [0.4151785746216777, 0.0] mm
```

```
2018-08-14 10:19:36,845 - numina.recipes.megara - DEBUG - nearest fibers
2018-08-14 10:19:36,845 - numina.recipes.megara - DEBUG - [311, 289, 333, 310, 313, 308, 314, 293, 329, 307, 316, 335, 312, 291, 331,
309, 315, 305, 317, 297, 325, 304, 319, 222, 400, 288, 334, 295, 327, 306, 318, 218, 404, 292, 330, 302, 320]
2018-08-14 10:19:36,846 - numina.recipes.megara - INFO - centroid: [0.3598269596081826, 0.05175193243532355] arcsec
2018-08-14 10:19:36,846 - numina.recipes.megara - INFO - centroid: [0.3212740713935877, 0.04620708253153888] mm
2018-08-14 10:19:36,846 - numina.recipes.megara - INFO - 2nd order moments, x2=0.356009, y2=0.356539, xy=-0.006142 arcsec^2
2018-08-14 10:19:36,846 - numina.recipes.megara - INFO - FWHM , x=1.405038, y=1.406084 arcsec
```

In this example, the brighest spaxel is located at `[0.4151785746216777, 0.0] mm` relative to the center of the field, which is, by definition located at `[0.0, 0.0] mm`. The positions of the centroids obtained from the 37 fibers around these spaxels are `[0.3212740713935877, 0.04620708253153888] mm` and `[0.260724285020397, 0.052190902407873864] mm`, respectively.

These centroid offsets (in mm), one or the other (to be decided by the user depending on the brightness of the target and on the presence of other bright targets in the field), are needed to derive recover all the flux from the standard star and to derive the instrument sensitivity curve for a particular setup using the *MegaraLcbStdStar* recipe.

In this case, the observation-result file was named *7_Standardstar.yaml* and includes spectrophotometric standard star individual exposures. The *"extraction_offset"* parameter can be computed as detailed in section 5.4.6 (this parameter is the same as in *6_lcbadquisition.yaml* for the same spectrophotometric standard star). The parameter *"reference_spectrum"* includes a text file where the flux-calibrated spectrum in AB magnitudes is provided[11]. This parameter can be also specify in the *control.yaml*. The *"reference_extinction"* parameter points to the text file with the information to apply the atmospheric extinction correction[12]. By default, the DRP searches for these data files in the `data/` directory. The *"position"* parameter is the position of the reference object, i.e. the offset in mm at the CCD detector computed with the recipe *MegaraLcbAcquisition*, written in the same format and units. Finally, the *"sigma_resolution"* parameter is the sigma of the Gaussian filter that would be used to degrade the resolution of the MEGARA input star spectrum. Given the high spectral resolution and low reciprocal dispersion of the MEGARA spectra this parameter is critical to remove artifacts associated to bright absorption lines present in the standard star spectrum, especially when the tabulated spectra have reciprocal dispersions that can be as high as 50 Å/pixel. The parameter *"ignored_sky_bundles"* contains the sky fiber ids to be ignored when the sky spectrum is computed (see more details in section 5.4.10 below).

```
(megara) bash-3.2$ more 7_Standardstar.yaml
id: 7_HR-R
mode: MegaraLcbStdStar
instrument: MEGARA
frames:
 - 0001286973-20170724-MEGARA-MegaraLcbImage.fits
 - 0001286974-20170724-MEGARA-MegaraLcbImage.fits
 - 0001286975-20170724-MEGARA-MegaraLcbImage.fits
requirements:
 extraction_offset: [+4.5]
 reference_spectrum: mbd284211_stis.dat
 reference_extinction: extinction_LP.txt
 ignored_sky_bundels: []
 position: [0.3212740713935877, 0.04620708253153888]
 sigma_resolution: 50
```

[11] The format of these files is the same as for those found in the ESO spectrophotometric standard stars database located at https://www.eso.org/sci/observing/tools/standards/spectra/.

[12] For processing standard-star observations this parameter must be defined in either the *control.yaml* of *7_Standardstar.yaml* files or the recipe would fail. In the case of the *MegaraLcbImage* or *MegarMosImage* recipes this would only imply that the processed data would not be corrected for atmospheric extinction.

Then the recipe is run by doing:

```
(megara) bash-3.2$ numina run 7_Standardstar.yaml -r ../control.yaml
```

Images provided in the observation-result file are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is present), bias and dark current (if *master_dark* is present) and corrected from pixel-to-pixel flat if *master_slitflat* is provided. The corrected images are then stacked using a median. The result of the combination is saved as an intermediate result, named *'reduced_image.fits'*.

The apertures in the 2D image are extracted, using the information in *master_traces.json* (or in the *model_map.json* if this file is present at the calibration tree) and the *"extraction_offset"* parameter set in the *7_Standardstar.yaml*, and then it is resampled according to the wavelength calibration in *master_wlcalib.json*. Then, the result is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named *'reduced_rss.fits'*.

The sky is subtracted by combining the 56 fibers dedicated for this purpose in the LCB mode. The RSS with the sky already subtracted is saved in a file named *'final_rss.fits'* as a result of the recipe. The flux of the star is computed by summing the signal in 37 fibers around the spaxel closest to the offset given in the *"position"* parameter so, finally, the *"star_spectrum"* is returned. This star spectrum is degraded with a Gaussian filter, corrected by atmospheric extinction and compared with the reference spectrum to return the *"master_sensitivity"*, which is finally stored at the `obsid_result/` directory.

```
(megara) bash-3.2$ tree obsid7_HR-R_work/ obsid7_HR-R_results/ -L 2
obsid7_HR-R_work/
├── 0001286973-20170724-MEGARA-MegaraLcbImage.fits
├── 0001286974-20170724-MEGARA-MegaraLcbImage.fits
├── 0001286975-20170724-MEGARA-MegaraLcbImage.fits
├── index.pkl
├── master_bias.fits
├── master_bpm.fits
├── master_fiberflat.fits
├── master_twilightflat.fits
├── reduced_image.fits
└── reduced_rss.fits
obsid7_HR-R_results/
├── fiber_ids.txt
├── final_rss.fits
├── master_sensitivity.fits
├── processing.log
├── reduced_image.fits
├── reduced_rss.fits
├── result.yaml
├── sky_rss.fits
├── star_spectrum.fits
└── task.yaml
```

The user can visualize the *master_sensitivity* curve running the python script *plot_spectrum.py* that can be found in the DRP distribution located at https://github.com/guaix-ucm/ (see

Figure 14).

```
(megara) bash-3.2$ cd obsid7_HR-R_results/

(megara) bash-3.2$ path_to_your_DRP_installation/megaradrp/tools/plot_spectrum.py  -s master_sensitivity.fits
```

The user needs to copy the file *master_sensitivity.fits* to the calibration tree at `ca3558e3-e50d-4bbc-86bd-da50a0998a48/MasterSensitivity/`.

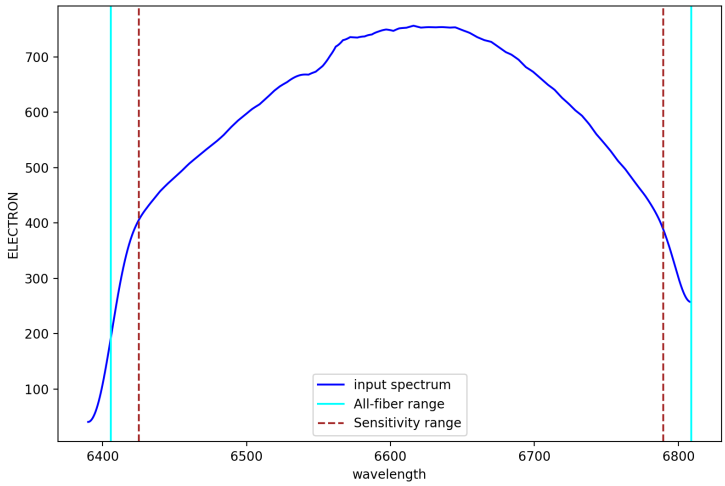


Figure 14: *Example of sensitivity curve for MEGARA HR-R VPH.*

5.4.10 LCB IFU/MOS scientific observation

Once all the calibrations files are derived and copied at the corresponding calibration directories, the user can reduce the corresponding scientific observations with recipes *MegaraLcbImage* or *MegarMosImage* depending on the observing mode (the LCB IFU or the MOS).

In this case, the observation-result files are named *8_reduce_LCB.yaml* for the LCB and *8_reduce_MOS.yaml* for the MOS mode, and include a list of all the frames obtained for the target. The *"extraction_offset"* parameter can be computed as detailed in section 5.4.6. The *"reference_extinction"* parameter can be provided here if it is not at the *control.yaml* file. The parameter *"ignored_sky_bundles"* contains the sky fiber ids to be ignored when the sky spectrum is computed. In the case of LCB observing mode, the dedicated sky-bundles have, by default, all ids in the range 93-100, that correspond to the following fibers (sorted in blocks of seven consecutive fibers starting from the ones of bundle 93, then bundle 94, 95, and so on, until bundle 100):

[22, 23, 24, 25, 26, 27, 28, 57, 58, 59, 60, 61, 62, 63, 134, 135, 136, 137, 138, 139, 140, 267, 268, 269, 270, 271, 272, 273, 351, 352, 353, 354, 355, 356, 357, 484, 485, 486, 487, 488, 489, 490, 561, 562, 563, 564, 565, 566, 567, 596, 597, 598, 599, 600, 601, 602]

In case MOS observing mode, sky-bundles should have been previously selected by the user for that purpose when preparing the observation with FMAT tool. If no sky-bundles are identified the DRP will not perform any sky subtraction to the target data.

The content of the *8_reduce_LCB.yaml* file is the following:

```
(megara) bash-3.2$ more 8_reduce_LCB.yaml
id: 8_HR-R_M15
mode: MegaraLcbImage
instrument: MEGARA
frames:
```

```
 - 0001309955-20170822-MEGARA-MegaraLcbAcquisition.fits
 - 0001309956-20170822-MEGARA-MegaraLcbAcquisition.fits
 - 0001309957-20170822-MEGARA-MegaraLcbAcquisition.fits
requirements:
 extraction_offset: [+6.5]
 reference_extinction: extinction_LP.txt
 ignored_sky_bundles: []
```

Then the recipe is run by doing:

```
(megara) bash-3.2$ numina run 8_reduce_LCB.yaml -r ../control.yaml
```

Images provided in the observation-result file are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is present), bias and dark current (if *master_dark* is present) and corrected from pixel-to-pixel flat if *master_slitflat* is provided. The corrected images are then stacked using a median. The result of the combination is saved as an intermediate result, named *'reduced_image.fits'*.

The apertures in the 2D image are extracted, using the information in *master_traces.json* (or in the *model_map.json* if this file is present at the calibration tree) and the *"extraction_offset"* parameter set in the *8_reduce_LCB.yaml*. These are then resampled according to the wavelength calibration in *master_wlcalib.json*. Then, the result is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named *'reduced_rss.fits'*.

The sky is subtracted by combining the 56 fibers (except the fibers listed in the *"ignored_sky_bundles"* parameter) dedicated for this purpose in the LCB mode. In case MOS observing mode, the sky is subtracted combining the signal of the fiber bundles (SKY bundles) selected by the user when preparing the MOS observation. The RSS with sky subtracted is saved in a file named *'final_rss.fits'* as a result of the recipe.

If a *master_sensitivity* is provided (optional), RSS products will be flux calibrated. If *reference_extinction* is provided (optional), *final_rss* and *reduced_rss* will be extinction corrected. Notice that *sky_rss* is not corrected for extinction.

```
(megara) bash-3.2$ tree obsid8_HR-R_M15_work/ obsid8_HR-R_M15_results/ -L 2
obsid8_HR-R_M15_work/
├── 0001309955-20170822-MEGARA-MegaraLcbAcquisition.fits
├── 0001309956-20170822-MEGARA-MegaraLcbAcquisition.fits
├── 0001309957-20170822-MEGARA-MegaraLcbAcquisition.fits
├── index.pkl
├── master_bias.fits
├── master_bpm.fits
├── master_fiberflat.fits
├── master_sensitivity.fits
├── master_twilightflat.fits
├── reduced_image.fits
└── reduced_rss.fits
obsid8_HR-R_M15_results/
├── final_rss.fits
├── processing.log
├── reduced_image.fits
├── reduced_rss.fits
├── result.yaml
├── sky_rss.fits
└── task.yaml
```

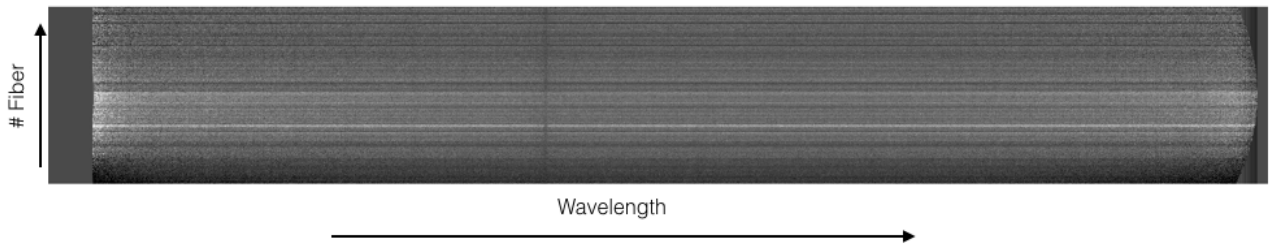


Figure 15: *Example of the final_rss.fits (sky subtracted, wavelength and flux calibrated) file for object M15 in the HR-R setup and the LCB observing mode.*

The following is an example of the products for M71 MOS observing mode data reduction:

```
(megara) bash-3.2$ tree obsid8_LR-R_M71_work/ obsid8_LR-R_M71_results/ -L 2
obsid8_LR-R_M71_work/
├── 0001288184-20170731-MEGARA-MegaraMosImage.fits
├── 0001288185-20170731-MEGARA-MegaraMosImage.fits
├── 0001288186-20170731-MEGARA-MegaraMosImage.fits
├── index.pkl
├── master_bias.fits
├── master_bpm.fits
├── master_fiberflat.fits
├── master_sensitivity.fits
├── reduced_image.fits
└── reduced_rss.fits
obsid8_LR-R_M71_results/
├── final_rss.fits
├── processing.log
├── reduced_image.fits
├── reduced_rss.fits
├── result.yaml
├── sky_rss.fits
└── task.yaml
```

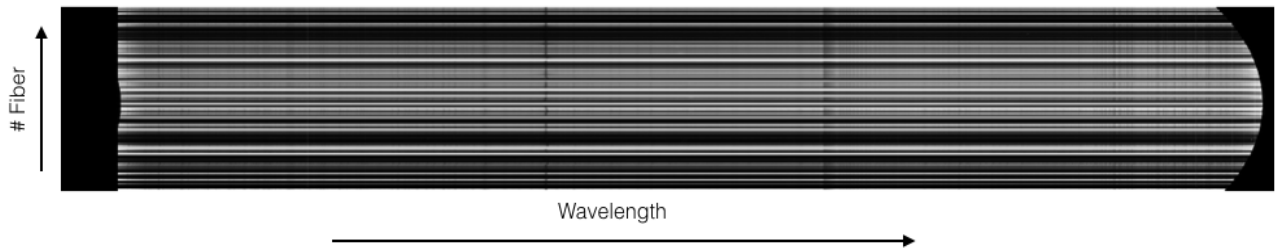


Figure 16: *Example of the final_rss.fits (sky subtracted, wavelength and flux calibrated) file for object M71 with the LR-R setup and the MOS observing mode.*

The user has also the option of running these recipes without performing any flux calibration. In order to do that one can simply add the following lines (shown in bold face below) in the corresponding .yaml file:

```
(megara) bash-3.2$ more 8_reduce_LCB.yaml
id: 8_HR-R_M15
mode: MegaraLcbImage
instrument: MEGARA
frames:
```

```
 - 0001309955-20170822-MEGARA-MegaraLcbAcquisition.fits
 - 0001309956-20170822-MEGARA-MegaraLcbAcquisition.fits
 - 0001309957-20170822-MEGARA-MegaraLcbAcquisition.fits
requirements:
 extraction_offset: [+6.5]
 reference_extinction: null
 master_sensitivity: null
 ignored_sky_bundles: [93,94]
```

6. KNOWN ISSUES

6.1 Matplotlib 3 in MacOSX:

In the case of using matplotlib 3 under MacOSX (installations starting in 2019), in order to avoid a potential error associated to the use of the libc++abi.dylib library by the TkAgg backend, we recommend you to add the following line to the $HOME/.matplotlib/matplotlibrc file (you should create the file if this does not exist):

```
(megara) bash-3.2$ more $HOME/.matplotlib/matplotlibrc
backend: TkAgg
```

Note that TkAgg is the default backend in MacOSX. If you want to use alternative ones they also need to be installed, or course.

6.2 Compiling in MacOSX Mojave:

Should the user be insterested in compiling part of the software on their own, we note that in the case of the Mojave distribution of MacOSX they should include the following environment variable in the $HOME/.bashrc shell configuration file:

```
(megara) bash-3.2$ more $HOME/.bashrc
export MACOSX_DEPLOYMENT_TARGET=10.9
```