SIPGI Documentation

Release 1.1.0

Adriana Gargiulo, Marco Fumana, Bianca Garilli, Paolo Franzetti, Susanna Bisogni, Marco Scodeggio

2021

CONTENTS:

1 Introduction **1**
- 1.1 SIPGI: VIPGI and LBT . . . 1
- 1.2 Overview . . . 2
- 1.3 How to use this manual . . . 2

2 How to Install SIPGI **3**
- 2.1 Linux . . . 3
- 2.2 macOS . . . 3

3 The SIPGI reduction cookbook **5**
- 3.1 Create a new Workspace and a new Project . . . 5
- 3.2 Files import . . . 6
  - 3.2.1 Add mask definition files for MODS MOS data . . . 6
  - 3.2.2 Raw frames import . . . 8
- 3.3 Bad Pixel Image . . . 8
  - 3.3.1 MODS Bad Pixel Image . . . 9
  - 3.3.2 LUCI Bad Pixel Image . . . 9
  - 3.3.3 Append Bad Pixels Image to data . . . 10
- 3.4 Create Master Files . . . 10
  - 3.4.1 Create Master Bias (MODS only) . . . 10
  - 3.4.2 Create Master Dark (LUCI only) . . . 10
  - 3.4.3 Pixel to Pixel Variation Image . . . 11
  - 3.4.4 Adjust First Guesses . . . 11
    - 3.4.4.1 Through-slit flat, lamp and scientific frames aligned . . . 11
    - 3.4.4.2 Lamp frames not aligned with through-slit flat and scientific frames . . . 14
    - 3.4.4.3 Through-slit flat and lamp frames not aligned with scientific frames . . . 15
  - 3.4.5 Master Flat . . . 15
  - 3.4.6 Utilities to check Master files . . . 15
    - 3.4.6.1 Show Spectra Location . . . 15
    - 3.4.6.2 Show Lambda Calibration . . . 16
  - 3.4.7 Master Lamp . . . 16
  - 3.4.8 Utilities to check the wavelength calibration . . . 17
    - 3.4.8.1 Check Lambda Calibration . . . 17
    - 3.4.8.2 Plot Lambda Calibration . . . 17
- 3.5 Preliminary Reduction . . . 18
- 3.6 Reduce Observations . . . 18
  - 3.6.1 Utilities to check the reduced observations . . . 20
    - 3.6.1.1 Show Reduced Spectra utility. . . . 20
    - 3.6.1.2 Manage Detections utility . . . 21

3.6.1.3 Plot Extraction Profile utility . . . 22
3.6.1.4 Show Window Table utility . . . 22
3.6.2 Molecfit . . . 22
3.7 Sensitivity function . . . 22
3.7.1 MODS . . . 23
3.7.2 LUCI . . . 23
3.7.3 Utility to check the sensitivity function: Plot Sensitivity . . . 24
3.8 Combine Observations . . . 24

4 Auxiliary system files **27**
4.1 Instrument files . . . 27
4.1.1 Paf files . . . 27
4.1.1.1 The Optical Model . . . 27
4.1.1.2 The Curvature Model . . . 28
4.1.1.3 The Inverse Dispersion Solution Model . . . 29
4.1.2 Grism tables . . . 30
4.1.3 CCD tables . . . 33
4.2 Calibration files . . . 33
4.2.1 Spectro-photometric standard . . . 33
4.2.2 Stellar templates . . . 34
4.2.3 Lines catalogs . . . 34
4.3 Mask files . . . 35
4.4 Extinction table . . . 36

5 Data organization **37**
5.1 Workspace . . . 37
5.2 Project . . . 38
5.3 Dataset . . . 38
5.4 Reduction Unit . . . 38
5.5 The importing procedure . . . 39

6 The SIPGI graphical interface **41**
6.1 The Starting and Setting Area . . . 41
6.2 The Data Manager Area . . . 43
6.2.1 Reduction unit layout . . . 44
6.2.1.1 Frames panel . . . 44
6.2.1.2 Master frames panel . . . 44
6.2.1.3 Generic master frames and calibration files . . . 45
6.2.1.4 Context menus . . . 45
6.3 The reduction and the analysis area . . . 46
6.3.1 The Reduction Tab . . . 46
6.3.2 The Analysis Tab . . . 46

7 The data reduction recipes **47**
7.1 Create Bad Pixels Image . . . 47
7.2 Append Bad Pixels Image . . . 48
7.3 Create Master Bias . . . 48
7.4 Create Master Dark . . . 49
7.5 Create Pix2pix Variation Image . . . 50
7.6 Adjust First Guesses . . . 52
7.7 Create Master Flat . . . 53
7.8 Create Master Lamp . . . 55

7.9 Preliminary Reduction . . . 57
7.10 Reduce Observations . . . 58
7.10.1 Refining the spectral tracing & wavelength solution on science frames . . . 58
7.10.2 Extraction of 2D spectra . . . 59
7.10.3 Sky subtraction . . . 60
7.10.4 Fringing Correction . . . 61
7.10.5 Flux Calibration & 1D extraction . . . 62
7.11 Create Sensitivity . . . 66
7.11.1 MODS sensitivity function . . . 67
7.11.2 LUCI sensitivity function . . . 67
7.12 Combine Observations . . . 69
7.12.1 Compute Offsets . . . 69
7.12.2 1D extraction . . . 70
7.13 Other data reduction utilities . . . 70
7.13.1 Science to Flat . . . 70
7.13.2 Science to Lamp . . . 71
7.13.3 Merge Lamps . . . 71
7.13.4 Append Table . . . 71
7.13.5 SIPGI To Molecfit . . . 71
7.13.6 Apply Molecfit Tra . . . 72
7.13.7 Revert Molecfit Tra . . . 72

8 The Analysis Utilities **73**
8.1 DS9 . . . 73
8.2 Primary Fits Viewer . . . 73
8.3 Secondary Fits Viewer . . . 73
8.4 Show Slit Position . . . 73
8.5 Show Spectra Location . . . 74
8.6 Show Lambda Calibration . . . 75
8.7 Check Lambda Calibration . . . 76
8.8 Plot Lambda Calibration . . . 77
8.9 Plot Sensitivity . . . 77
8.10 Show Reduced Spectra . . . 77
8.11 Manage Detections . . . 78
8.12 Plot Extraction Profile . . . 78
8.13 Show Window Table . . . 79

9 Appendix **81**
9.1 Appendix A: the stellar templates . . . 81
9.1.1 The Pickles templates . . . 81
9.1.2 The spectro-photometric standards . . . 82
9.2 Appendix B: keywords necessary to SIPGI to import and categorize raw data . . . 82
9.2.1 LUCI . . . 82
9.2.2 MODS . . . 84
9.3 Appendix C: error propagation on 1D and 2D spectra . . . 84
9.3.1 Error on preliminary reduced frames . . . 84
9.3.2 Error on reduced frames . . . 85
9.3.2.1 Error for un-distorted slits in reduced frames. . . . 85
9.3.2.2 Error for distorted slits in reduced frames. . . . 86
9.3.3 Error on combined frames . . . 87
9.4 Appendix D: 2D spectra extraction and resampling . . . 87
9.5 Appendix E: bad pixels correction and cosmic rays cleaning . . . 89

10 Glossary 91

11 References 93

CHAPTER

ONE

INTRODUCTION

The Spectroscopic Interactive Pipeline and Graphical Interface (SIPGI) is an evolution of the VIMOS Interactive Pipeline and Graphical Interface (VIPGI). VIPGI is a complete data reduction environment, originally designed to carry out the reduction of spectroscopic data acquired with the VIMOS spectrograph at the European Southern Observatory (ESO) Very Large Telescope. VIPGI was designed for the reduction of the VIMOS VLT Deep Survey (VVDS) data, but its capabilities were sufficiently general to be easily adaptable to a variety of VIMOS data (Scodeggio et al. 2005, Garilli et al. 2012). During the years, the VIPGI efficiency and the quality of its data reduction products were such to make VIPGI the reduction pipeline of the major extragalactic surveys carried out with VIMOS: VVDS, zCosmos, VUDS, VIPERS, VANDELS for a total of more than 200000 spectra fully reduced and calibrated.

1.1 SIPGI: VIPGI and LBT

At the core of VIPGI (and SIPGI as well) is the Data Reduction Software (*DRS*), i.e. the set of programs and methods developed for the main steps of the data reduction.

The DRS is designed to be adaptable to other instruments. About ten years ago, SIPGI was born tuning the VIPGI core to make it able to reduce the spectroscopic data of the two main Large Binocular Telescope (LBT) spectrographs: MODS and LUCI.

Similarly to VIMOS, MODS is an optical spectrograph, so the main changes concerned the peculiarities of MODS data but not the deep core of the DRS. LUCI is a near infrared spectrograph and its different wavelength domain required deep revisions of the DRS. SIPGI has been used by the LBT spectroscopic data reduction center located at INAF-IASF Milan to reduce all the MODS and LUCI spectra acquired during the Italian time in the last ten years.

The documentation presented here is a cookbook to reduce MODS and LUCI spectra with this customized SIPGI version. The documentation is surely not complete, nor exhaustive of all SIPGI functions, features (and bugs). We rely on users feedback to improve it in the next versions.

For any question, comment, suggestion or for any request of help in the reduction of MODS/LUCI spectra, please contact lbt-italia-spec@inaf.it

1.2 Overview

SIPGI is composed by two main parts: the built-in data organizer and the DRS.

Although the data reduction recipes in DRS automate to a very large extent the task of reducing spectroscopic data, it does not address at all two important and problematic areas of the global data reduction activity: the organization of the (large) volume of the data and the validation of input data for reduction recipes. Each reduction recipe works under the assumption that the correct input (in term of calibration and scientific data) is provided to it, and could produce totally unpredictable results if this assumption were not to be met. To address these two points SIPGI is provided with a built-in data organizer that helps to manage the data and to provide the various recipes with the correct input. The organization of files is not optional in SIPGI. To be processed by the reduction recipes, raw FITS files **must** be ingested by the built-in data organizer.

Once files are properly digested and classified, all the recipes of the DRS can be used and their data products can be inspected using the analysis utilities (see *Analysis Utilities*).

1.3 How to use this manual

This manual is organized as follow: in Chapter 2 all the information necessary to install the software is provided, while in Chapter 3 a road-map for the data reduction of MODS and LUCI spectra with SIPGI is presented. Following the instructions provided in this Chapter, the user should hopefully be able to fully complete a data reduction. No information on the SIPGI basic concepts and/or on the SIPGI recipes is provided in Chapter 3. The user can retrieve this information by following the links that are embedded in the text.

In the following chapters a fully description of the software is provided. In Chapter 4, the SIPGI Auxiliary system files, i.e. a fundamental part of the SIPGI world, are presented; in Chapter 5, the functioning of the built-in data organizer is shown; in Chapter 6, a complete description of the graphical interface is provided; in Chapter 7, all the DRS recipes are described, and finally in Chapter 8, all the Analysis utilities are explained.

Although the cookbook in Chapter 3 should allow user to complete his/her data reduction, sometimes the spectra reduction may not be such a straightforward path due to data peculiarities. In this case, a complete view of what is the software and how it works could help in finding the best solution.

To summarize, we made all our efforts to allow users to reduce their LBT data skipping the boring manual part related to software functioning, but for a conscious use of SIPGI, it is strongly recommended to read Chapter 4,5,6,7,8 *before* Chapter 3 and *before* starting a data reduction.

Note: In this manual, we try to provide all the support for a quick and easy reduction, however we assume that the user handles the basic concepts behind the reduction of spectroscopic data.

CHAPTER

TWO

HOW TO INSTALL SIPGI

SIPGI is distributed in binary form for Linux and macOS systems as part of the PANDORA SUITE. The suite can be downloaded from the Pandora website

2.1 Linux

We distribute our code compiled on two Linux flavours. The CentOS 7 distribution is to be used on older systems (having a GLIBC version >= 2.17), while the Xubuntu 18.04 should work on newer ones (having a GLIBC version >=2.27). Download the installer script for your OS version and execute it:

shell> bash Pandora-1.0-CentOS7.sh

Follow the on-screen instructions and install the suite in a directory of your choice.

Most of the software dependencies are included into the distribution. However a few of them are "system related" and must be met by the OS on which you are installing our code; in particular our code relies on the mesa-GLU libraries that are not always installed by default. A check is made at the end of the installation; in case of error check if the mesa-GLU libraries are installed or contact us at lbt-italia-spec@inaf.it.

Once installed SIPGI can be launched using the command *<installation-dir>/bin/sipgi*.

2.2 macOS

Download the DMG file suitable for you macOS version and open it. Double-click on the *Pandora.pkg* file and the suite will be installed into the */Applications/Pandora* directory. Depending on the security level of your system an **"unidentified developer"** error can appear; in this case right-click on the *Pandora.pkg* file and select *Open* to override this issue.

Once installed SIPGI can be launched in two ways:

- from the command line with the command */Applications/Pandora/bin/Sipgi*
- using the macOS app located in */Applications/Pandora*

CHAPTER

THREE

THE SIPGI REDUCTION COOKBOOK

The reduction procedure within SIPGI consists in the following steps:

- *new workspace and a new project creation*;
- *raw data import* into SIPGI;
- *bad pixel map creation*;
- *master calibration files creation* (Master Dark, Master Flat and Master Lamp), both for science and standard/telluric frames;
- *pre-reduction* of scientific and standard/telluric frames with a proper Master Flat (and optionally Master Dark and Master Bias);
- *reduction* of standard/telluric frames with the proper Master Lamp, if flux calibration is desired;
- *sensitivity function creation*, if flux calibration is desired;
- *reduction* of scientific frames with the proper Master Lamp (and sensitivity function if flux calibration is desired);
- *combining* of all the reduced scientific frames to obtain a combined frame containing the final spectra for each object in the mask.

Each step is described in detail in the following sections.

3.1 Create a new Workspace and a new Project

Before starting with a new data reduction in SIPGI, the user must group the raw files into a directory (e.g. /home/user/raw_data, preferentially on internal disk). Ingestion of raw files in SIPGI from external hard disk takes considerably longer time.

Launch SIPGI and before creating a new *Workspace* and a new *Projects*, the user has to set some general preferences. Click on the `SIPGI->Preferences` menu in the *Starting and Setting Area* and set the primary (see *Primary Fits Viewer*), and the secondary FITS viewer (see *Secondary Fits Viewer*) as well as the log level desired (see the *log level note*).

Open the workspace creation panel selecting the `Workspace->New` menu in the *Starting and Setting Area*; in the panel insert the `Workspace name` (e.g. MODS_reduction) and select the `Instrument`. If a *Workspace* for the same instrument already exists, the user can copy the setting parameters from this *Workspace* using the `Copy setup from` entry.

Click on the `Create` button and the new *Workspace* is created.

In the *Workspace* just created, open the project creation panel selecting `Project->New` menu in the *Starting and Setting Area*. In this new panel define the `Project name` (e.g. BLZ_reduction) and insert an existing and empty `Project directory` (e.g. /home/user/BLZ_reduction), that is the directory where all the SIPGI imported files (see below) and pipeline reduction products will be stored. This directory must exist before the creation of a new Project (i.e. SIPGI will not create a directory with the name the user gives to the Project), and it **must not** be the raw files directory.

3.2 Files import

To be processed by SIPGI, raw FITS files must be ingested by the built-in data organizer (see *Data organization*). During the importing procedure, SIPGI creates a *copy* of raw files in the Project directory, and - among other things - rotates and flip the data and appends to them all the necessary auxiliary system files (for more details see *The importing procedure*).

Important: During the import procedure, SIPGI creates a **copy** of raw files. Before starting the ingestion, the user must check to have enough space on his/her disk for this copy.

3.2.1 Add mask definition files for MODS MOS data

MODS MOS frames **do not contain** information about mask geometry. This information is fundamental to locate the 2D spectra on the frames and it is required by the importing procedure. If data to import are MODS MOS files, the *mask file* **must** be added to the auxiliary files before the data import starts. Open the `Custom Auxiliary Files Manager` clicking on the `Custom Auxiliary Files` button at the bottom of the *The Data Manager Area*. Click on `Add Auxiliary File` button. Upload the mask file in the `Auxiliary File` panel and fill in all the entries. Please, take care to insert the `Mask name`. This field **must** match the MASKNAME keyword contained in the FITS header of the raw files to import (see Fig. 3.1).

Note: The `Add Auxiliary File` button allows the user to add all his/her customized auxiliary files (e.g. customized *stellar template*, customized *Spectro-photometric standard*, customized *Lines catalogs*, etc…). Once a new file is uploaded, it is ingested in SIPGI that can use it for reduction purposes. The `Edit File`, `Rename File` and `Delete File` buttons in the *Custom Auxiliary Files Manager* window allows user to edit the specifications of the selected customized auxiliary file, to rename this file or to delete it from SIPGI. In addition to this, the `View/Hide System Files` allows the user to see all the auxiliary system files. The `FileType` drop-down menu at the top of the *Custom Auxiliary Files Manager* window permits to select the type of auxiliary files to show in the central panel.

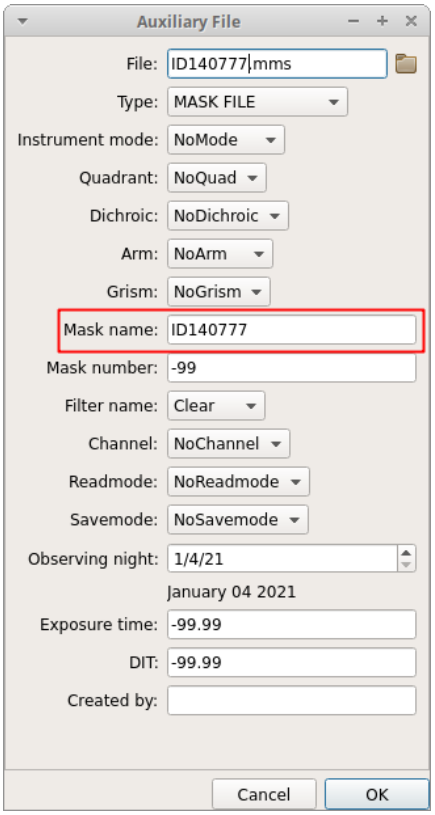

Fig. 3.1: In figure is reported an example of the `Auxiliary File` panel for the upload of a mask file. Red box indicates the Mask name. This entry is mandatory and it must be filled with the same MASKNAME keyword in FITS header of the raw files.

3.2.2 Raw frames import

Note: SIPGI imports only not compressed FITS files.

Clicking on the `Import` button at the bottom of the SIPGI window, the `ImportRawFiles` window appears. The `Select directory` button allows user to select the directory with raw files (e.g. /home/user/raw_data) and import *all of the FITS files* it contains; the `Select Files` button allows user to select specific files to import.

Clicking on the `Start Import` button, SIPGI starts the ingestion of the raw files processing them with the built-in data organizer (for more details see the *The importing procedure*).

Caution: The ingestion of raw data can take a while depending on the number of files to import. Look at the *log* to follow the import execution. **Please, do not stop the import before the process is finished.**

The import procedure creates a number of *Datasets* and *Reduction Units* listed in the main panel of the data manager area.

Raw files import is an incremental procedure; new raw data can be imported as they become available. For example, if data were acquired in different nights, the user can ingest and reduce the data of the first nights and then add other files as they arrive. The exposure sequence number provided by SIPGI (see the *renames the file* paragraph) to these new imported files follows the sequence of files already ingested.

Note: Data of the same target observed in different periods could have different OBJECT keyword (e.g. due to a different target name in the Observing Blocks). The import procedure will split these data in different *Datasets*. To unify the data in the same *Dataset* and reduce them together, click on the *Datasets* with the right button of the mouse, and rename them with the same name.

SIPGI will not digest and categorize raw files with incorrect entries for the keywords listed in *Appendix B*. The files not correctly imported will be stored in the `Unclassified` panel in the *Data Manager Area*.

3.3 Bad Pixel Image

The first step in the reduction process is to create the bad pixel image. The *Create Bad Pixel Image* recipe is able to automatically detect 2 kinds of bad pixels: dead pixels and hot pixels. Slitless flats are used to detect dead pixels and dark frames are used to detect hot pixels. If just one type of file is provided, the bad pixel image will store only the relative bad pixels.

The created bad pixel image will be available in the *Master Frame Panel* of the current *Reduction Unit*.

3.3.1 MODS Bad Pixel Image

Select the *Reduction Unit* with slitless flats. Check the parameter values of the `Create Bad Pixels Image` recipe going into `Parameter Files->Create Bad Pixel Image` menu in the *Starting and Setting Area*. To run the recipe, click on the `Run` button. A new window appears. Select the slitless flats to be used in the *Reduction Unit* and then click on the `Add files` button in the recipe pop-up window.

Caution: Take care to use correct input files, e.g. MODS1 slitless flats for the bad pixel map of MODS1 data. The drop-down filtering menu on top of *Frames Panel* of the *Reduction Unit* can help in the selection of proper inputs.

Click the `Run Recipe` button. A new pop-up window will appear with a suggestion for bad pixel image name. A personal name can be provided. It is good practice to give a name that preserves the filter/dichroic name (e.g G400L, G670L, G400L_Dual, G670L_Dual) and the LBT arm (e.g. MODS1, MODS2). This because, e.g., the bad pixel map for MODS1 and MODS2 will be stored in the same place (i.e. the *Generic Master Frames and Calibration Files* of the *Project*, see *Append Bad Pixels Image to data*) and a clear name can help in distinguishing them. The pop-up window allows also to overwrite an existing bad pixel map with the same name.

Check the result viewing it with one of the two FITS viewer selected and, if it is not satisfactory, modify the recipe parameters and re-launch the recipe.

The user can save the customized recipe setting parameters using the `Save` button in the parameters panel. If the user is satisfied with the parameters stored, he/she can run the recipe directly from the `Create Bad Pixels Image` button in the `Reduction` tab on the left.

3.3.2 LUCI Bad Pixel Image

To create bad pixel image for LUCI data, the procedure is the same as MODS data with the possibility to also load dark frames to identify hot pixels. Once slitless flat frames have been uploaded in the DEAD PIXELS FRAMES list panel, dark frames can be uploaded in the HOT PIXELS FRAMES list selecting them from the appropriate *Reduction Unit*.

Important: Once the bad pixel map is created, it must be promoted from the *Master Frame Panel* of the *Reduction Unit* in which is created to the *Generic Master Frames and Calibration Files* of the *Project*. This operation will make this data product available to other *Reduction Units* of the *Project*. To do this, click on the bad pixel map with the right mouse button and select `Share as auxiliary` from the context menu. This will move the bad pixel map from the *Master Frame Panel* to the *Generic Master Frames and Calibration Files*. If the user wants to delete/rename a file in the Generic Master Frames and Calibration Files panel, she/he must click on the `Custom Auxiliary Files` button, must select the file and perform the desired operation (see this *note* for more details.).

3.3.3 Append Bad Pixels Image to data

Once the user has a satisfactory bad pixels image, he/she must *append it to data*. This allows the reduction recipes to have information on the pixels to be corrected for each frame.

Check the parameter values of the `Append Bad Pixels Map` recipe going into `Parameter Files` menu. Select all the data to polish, the bad pixel map itself and click the `Append Bad Pixels Image` button in the `Reduction` Tab.

The bad pixel map must be appended to all the frames that user wants to clean up.

3.4 Create Master Files

The Master calibration frames are: Master Bias, Master Dark, Master Flat and Master Lamp. Together with the bad pixel maps and sensitivity functions, the master files are those files that are applied to all the scientific frames.

As for the bad pixel map, after creating the Master frames they appear in the *Master Frame Panel* of the *Reduction Unit* in which are created. In order to use them, the user must promote these file to the *Generic Master Frames and Calibration Files* (see the *note*).

3.4.1 Create Master Bias (MODS only)

Master Bias is used to subtract bias level from MODS science frames.

The creation of a Master Bias is not mandatory in SIPGI. Actually standard practice for MODS data is to subtract the bias level using the prescan/overscan region. If the user is satisfied with this choice, he/she can skip the creation of the Master Bias.

If the user wants to subtract bias level using bias frames, he/she must *create a Master Bias*. Check the parameter values of the `Create Master Bias` recipe going into `Parameter Files`. After this, select the bias frames and click the `Create Master Bias` button.

Caution: Frames used to create the Master Bias **must** have the same ARM and READMODE of the science frames (e.g. 8kx3k for MODS data, see MODS manual).

A pop-up window appears asking for the Master Bias name and the possibility to overwrite an existing file with the same name. It is good practice to give a name that preserves the the LBT arm. `Run` button launches the recipe.

3.4.2 Create Master Dark (LUCI only)

Master Dark is used to remove the dark current level plus bias level from LUCI science frames.

To *create a Master Dark*, check the parameter values of the `Create Master Dark` recipe going into `Parameter Files` menu. If the `CLEAN_BAD_PIX` keyword is activated, append the appropriate bad pixel map to dark frames (see *Append Bad Pixels Image to data*).

The user has to select the dark frames and click the `Create Master Dark` button.

Caution: Take care to use dark frames with the same ARM, DIT and READMODE of science frames to reduce.

A pop-up window appears asking for the Master Dark name and the possibility to overwrite an existing file with the same name. It is good practice to give a name that preserves the READMODE (e.g MER or LIR) and the LBT arm. `Run` button launches the recipe.

3.4.3 Pixel to Pixel Variation Image

This image is used to correct for the pixel-to-pixel variation in raw frames. The recipes combines together input slitless flats and removes from this combined image the large scales fluctuations. The resulting image only contains the pixel-to-pixel variation (see *Create Pix2pix Variation Image* for more details). This step is not mandatory.

The first step to create a Pixel to Pixel Variation Map is to check the parameter values of the `Px2Px Variation Image` recipe going into `Parameter Files` menu. If the `CLEAN_BAD_PIX` keyword is activated, append the appropriate bad pixel map to the slitless frames (see *Append Bad Pixels Image to data*). After this, select slitless flats and click the button `Create Px2Px Variation Image`. A pop-up window appears asking for the Pixel to Pixel Variation Image name and the possibility to overwrite an existing file with the same name. It is good practice to give a name that preserves LBT arm. `Run` button launches the recipe.

3.4.4 Adjust First Guesses

The Reduction recipes in SIPGI rely on a set of models which describe the spectra location on the CCD (*OPT Model*), the spectra curvature (*CRV Model*) and the inverse disperse solution (*IDS Model*). The coefficients of these models are stored in the *PAF files*. These models provided with SIPGI and appended to raw files are estimated on the basis of instrument characteristics, but may not be exactly representative of the current data because of, e.g. changes in the optical distortions on a night base or masks not perfectly aligned in the FPU. For this reason, they must be checked and if needed, re-adjusted. This *interactive task* allows to control the *Paf file* stored in the input through-slit flat on the input lamp frame.

The first step in this contest is to verify whether calibration files (e.g. through-slit flats and lamps) are aligned with scientific frames (see the Attention note in the *Adjust First Guesses* recipe). This can be easily done opening a flat, a lamp and a scientific frame with ds9 and selecting in it `Frame->Frame Parameters->Tile->Rows`.

3.4.4.1 Through-slit flat, lamp and scientific frames aligned

Select a *Reduction Unit* with the through-slit flat and click on the flats one desires to use for the Master Flat creation. Then click the `Adjust First Guess` button in the `Reduction` Tab. A new panel opens, showing on the left the list of flat fields selected. Browse in the *Reduction Unit* containing the lamp frames with the same grism/filter/dichroic/arm of the flats and select one of them. Click on the `Add lamp files` in the `Adjust First Guess` panel.

The `Adjust Grism Table` button permits to see and edit the *Grism tables*. This task allows the user to check and adjust the *Grism Table*. As example, in LUCI data with central wavelength different from the nominal one, the user can update the central wavelength (and extraction limits accordingly). Optionally, new sky lines can be added to the *Grism Table* with the the `Add Skyline` button.

Once the *Grism Table* is set, choose one through-slit flat and one lamp frame in the panel clicking on them. Then start the adjust procedure selecting the `Adjustment type` and clicking the `Run` button. It is strongly recommended to start with `Shift only` adjustment type, and then continue with `Rotate` and `Complete Adjust` type.

Note: Standard observations provide user with different lamp frames. Each lamp has their own emission lines. In order to have a better sampling of the wavelength solution it could be useful to merge lamps together. To do this, before running the Adjust First Guess task, select all the different lamps, the suitable *Lines catalog* from the *Generic Master Frames and Calibration Files* and then click the `Merge Lamps` button in the `Reduction` tab. A pop up window appears asking for the name of the merged files. Click `Run`. The resulting sum file is shown in the lamp *Reduction Unit*. The user can provide this file as input to the Adjust First Guess task.

Shift first guesses

This step corrects the *OPT model* offset adjusting the X and/or Y shifts of the first guesses with respect to data in use. Select Shift Only as `Adjustment type`, and push the `Run` button.

The selected lamp frame is displayed in a new ds9 window, with some green regions superimposed that show the first guesses of the spectral tracing and of the wavelength calibration following the *Paf file* stored in the input flat.

The green vertical lines indicate where the left edge of the spectrum is supposed to be, and the horizontal lines show where the arc lamp lines (those in the *Lines catalog* appended to the lamp frame and within the extraction limits) should be. Clicking on one of the blue rectangles, and moving it, all such regions will move accordingly.

In this phase the goal is to move green lines (using the blue rectangle) in order to match just the left edge of the central slit and the position of the most central lines. The center of the green horizontal lines should be positioned roughly on the center of the corresponding frame lines. It could happen that edges of other lateral spectra and/or other arc lines are still displaced but this will be fixed in the next steps. When satisfied, click `Recompute` on the question panel. The recipe recomputes the OPT Model using these refined input constraints and will update the green region on the image. The user must check that the new models have stored the changed suggested. If this is not the case, the user can iterate the procedure till he/she is satisfied with the newly computed first guesses, and then exit the loop by clicking on `Done`.

Attention: Save the first guesses clicking on the `Save` button on the Adjust First Guess panel. This will update the *Paf file* in the selected flat file.

Rotate

Distortions produce curved spectra, and the *CRV Model* is the model which describes this curvature. This step adjusts the first coefficient of the curvature model (the linear term). Choose Rotate as `Adjustment type`, and push `Run`.

The lamp image is displayed in ds9; the green vertical line shows the position of the edge of the spectrum according to the updated *Paf coefficients*. Click on the top end or on the bottom end of the green line to rotate it such that it follows the edge of the spectrum. If necessary, the user can move the vertical line near the edge of the spectrum to facilitate the tracking.

When satisfied, click `Recompute` on the question panel. A new CRV Model is computed, and the result is shown on the same image. It is possible to iterate the procedure, or exit the loop by clicking on `Done` when the newly computed first guesses are considered satisfactory.

Attention: Do not forget to push the `Save` button to store changes into the *Paf file* in the input flat file.

Complete adjustment

The complete adjustment is strictly related to the lambda calibration of the spectra and it corrects the *IDS Model*. Choose Complete Adjustment as `Adjustment type` and push `Run`.

The lamp image is displayed on ds9 again; the green lines show where the arc lamp lines should be according to the IDS model stored into the flat *Paf file*.

By clicking on one green line, you can select it and move it so that it falls exactly on the corresponding arc line. The scope of the game in this phase is to have all the green lines as near as possible to the real corresponding arc lines. All the shifts in the x directions will be ignored. It is possible to select more than one region by multiple selection facilities provided by ds9 and move them solidly. A red region with ALL lines is also shown, to help the identification of very close lines (the user can use it as a ruler and move such red region around, to help the eye in the most difficult cases). When satisfied, click `Recompute` on the question panel. A new set of first guesses is computed, and the result shown on the same image. It is possible to iterate the procedure, or exit the loop by clicking on `Done` when the newly computed first guesses are considered satisfactory.

In this step, the user can activate the entry `View lines label` which will show for each displayed line its wavelength.

Important: In the Complete Adjustment:

- the user **can** delete green lines to exclude them from the fit;
- do not mismatch lines;
- push `Recompute` only when all slits of the whole mask are in the proper position. The recipe computes a global model and it will fit all the slits; a partially adjusted mask produces a no-sense fit and all the work will be lost;
- In case some green lines fall out of the frame (too blue or too red) during the adjustment, they can be deleted and would be excluded from the fit;

- in LS data, the curvature of emission lines can be so important that the straight green lines can intercept two close emission lines. This can lead to non accurate wavelength calibration. In this cases the user can split the line in `Split slit` part (see Fig. 3.2).

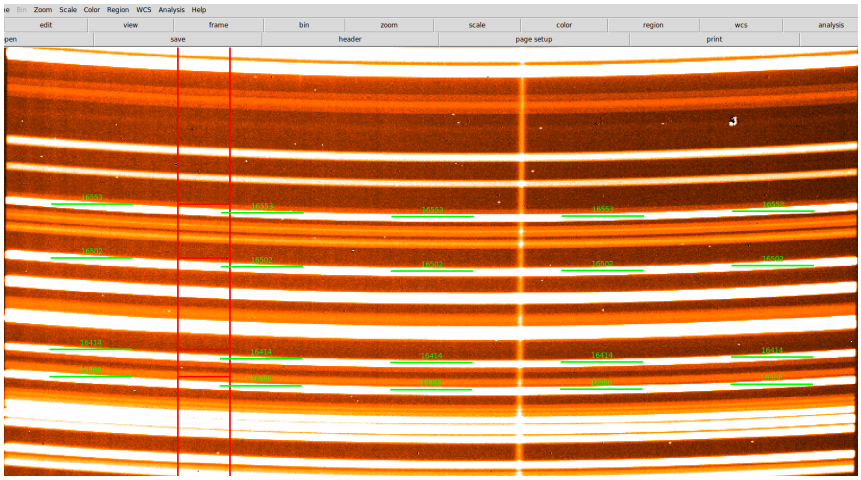

Fig. 3.2: The figure shows the behavior of the Complete Adjustment if `Split slit` =5.

Once the user is fully satisfied with the first guesses, he/she can save the first guesses pushing on the `Save` button.

Note: In the last step it is important to save the new *Paf files* in all of the through-slit flat of the list. To do this, the user can choose *All frames* in the menu `Save results into`.

3.4.4.2 Lamp frames not aligned with through-slit flat and scientific frames

In case that through-slit flat and science frames are aligned with each other but not with the lamp frames, the user can use science frames as lamp. In this case the spectral tracing is performed directly on the science frame bypassing the issue of the shift, and the wavelength calibration is performed on sky lines.

To do this, the user must convert a science frame into a lamp frame using the `Science to Lamp` button in the `Reduction` Tab (see *Science to Lamp*). Select a raw science frame and a suitable sky-line catalog from the *Generic Master Frames and Calibration Files* and click on the `Science to Lamp` button. This creates a new file (hereafter science-lamp frame) with the same name of the science frame plus the prefix *lp*. This new frame will appears in the same *Reduction Unit* of science frames.

Once the science-lamp is created, the user can use this file as lamp in the adjust procedure following the same steps described in the previous section.

3.4.4.3 Through-slit flat and lamp frames not aligned with scientific frames

If neither through-slit flats nor lamp frames are aligned with science frames, the user must use science frames both as flat and as lamp. As in the previous case, the user must use a science frame as lamp in order to define suitable first guesses for the spectral tracing. However if through-slit flats are shifted with respect to science frames, it makes no sense to use them as inputs in the Create Master Flat recipe. To overcome this limit, the user can convert science frames in flat frames using the `Science to Flat` button in the `Reduction` Tab (see *Science to Flat*). Select a raw science frame and click on the `Science to Flat` button. This creates a new file with the same name of the science frame plus the prefix *ff* (hereafter science-flat frame). This new frame will appear in the same *Reduction Unit* of science frames.

Once the science-lamp and the science-flat are created, the user can use these files in the adjust procedure following the same steps described in previous section. Clearly, in this case, Master Flat will not include the pixel-to-pixel variation map, unless a Pix2Pix Variation Image is provided (see later).

3.4.5 Master Flat

The creation of a Master Flat is mandatory in SIPGI. It contains two information: the location of 2D spectra on the CCD and the pixel-to-pixel variation.

The pixel-to-pixel variation is computed by the *Create Master Flat* using the input through-slit flats or using the *Pix2Pix Variation Image*, if provided.

If the user is reducing MODS data, it is strongly recommended to use a dedicated Pix2Pix Variation Image in order to take into account the different gains between quadrants. This applies also with not-aligned science and through-slit flat frames.

To create a Master Flat, check the parameter values of the `Create Master Flat` recipe going into `Parameter Files` menu. If the `CLEAN_BAD_PIX` keyword is activated, append the appropriate bad pixel map to the through-slit flat frames (see *Append Bad Pixels Image to data*). Select the through-slit flat frames (or science-flat frames) to combine, the optional Pix2Pix Variation Image and the optional Master Bias if `MASTER` subtracting method has been selected (only for MODS reduction). Click on the `Create Master Flat` button. A pop-up window will appear asking for the Master Flat name and the possibility to overwrite an existing file with the same name. It is good practice to give a name that preserves the the LBT arm. `Run` button will launch the recipe.

A Master Flat is created and listed in the *Master Frame Panel* of the *Reduction Unit*. The user must promote it to the *Generic Master Frames and Calibration Files*.

As stated in *Create Master Flat*, the Master Flat takes into account for the pixel-to-pixel variation, but not for large scales variations.

3.4.6 Utilities to check Master files

3.4.6.1 Show Spectra Location

SIPGI offers the possibility to check the Master Flat with the utility `Show Spectra Location` in the `Analysis` Tab. For more details on its functioning see the *Show Spectra Location* description.

Select one of the through-slit flats used to create the Master Flat and the Master Flat itself. Click on the `Show Spectra Location` button. A ds9 window appears showing the through-slit flat with the fitted tracing highlighted (see *Show Spectra Location* for more details).

A Master Flat is considered satisfactory when the tracing follows the spectra edges. This condition is necessary for a proper extraction of 1D spectra.

Note: Check that the extraction limits (indicated with bold black "lines" in the bluest and reddest part of the spectrum) are within the tracing limits (white vertical "lines", see *Show Spectra Location* for more details).

The utility `Show Spectra Location` can be used also on science frames. This allows user to directly check the spectral tracing on science raw frames.

3.4.6.2 Show Lambda Calibration

Another utility SIPGI provides to check the Master Flat (and Master Lamp) is the `Show Lambda Calibration` in the `Analysis` tab (see *Show Lambda Calibration* for more details). The `Show Lambda Calibration` utility allows to check the IDS model stored in the input Master Flat.

Select one of the lamp or science-lamp frame and the Master Flat. Click on the `Show Lambda Calibration` button. A ds9 window will appear showing the input frame.

A satisfactory Master Flat has the first guesses of the IDS model (indicated by thin solid horizontal lines) that provides good prediction of the lamp lines position.

3.4.7 Master Lamp

The creation of a Master Lamp is mandatory in SIPGI. The Master Lamp is the frame which stores in the EXR table extension the best wavelength solution.

The *Create Master Lamp* recipe starts from the first guesses of the wavelength solution stored in the EXR extension of input Masters Flat and refines the models on real data. The best-fitted solution is stored into the EXR table of the Master Lamp.

The recipe works both on real lamp frames and on science-lamp frames (see *Lamp frames not aligned with through-slit flat and scientific frames*).

To create a Master Lamp, check the parameter values of the `Create Master Lamp` recipe going into `Parameter Files` menu. If the `CLEAN_BAD_PIX` keyword is activated, append the appropriate bad pixel map to input frames (see *Append Bad Pixels Image to data*).

Select the desired input frame(s) (lamp frames or science-lamp frames), select the appropriate Master Flat and the optional Master Bias if `MASTER` subtracting method has been selected (only for MODS reduction). Click on the `Create Master Lamp` button. A pop-up window will appear asking for the Master Lamp name and the possibility to overwrite an existing file with the same name. It is good practice to give a name that preserves the the LBT arm. `Run` button will launch the recipe.

A Master Lamp is created and listed in the *Master Frame Panel* of the *Reduction Unit*. The user must promote it to the *Generic Master Frames and Calibration Files*.

3.4.8 Utilities to check the wavelength calibration

Once a Master Lamp is created, the user can visually check the result using the *Show Lambda Calibration* utility.

Select one of the lamp or science-lamp frame and the Master Lamp. Click on the `Show Lambda Calibration` button. A ds9 window will appear showing the input frame. If no input frame is provided, the utility displays the solution directly on the master frame image.

In this check the user must control that all the thin horizontal "lines" are placed on the relative emission lines and that they are fairly continuous. Since the wavelength solutions are computed slice by slice (see *Create Master Lamp* for more details), a one-pixel shift between the points of each thin "line" is allowed due to interization issue. Fluctuations bigger than one pixel indicate a poor calibration in that spectral region.

Another useful check is to look at the black bold lines which delimit the extraction range. The Create Master Lamp recipe derives the wavelength solution using all the input lines. This solution is then extrapolated up/down to the red/blue extraction limits. The `Show Lambda Calibration` utility will show also the expected position of the extraction limits. If the extraction limits "lines" are blurred, it means that the best-fit wavelength solution is poorly constrained in the region between the bluest/reddest calibration line and the relative extraction limit.

3.4.8.1 Check Lambda Calibration

Once the visual inspection is considered satisfactory, the user can quantify the quality of the wavelength calibration using the utility `Check Lambda Calibration` in the `Analysis` tab (see *Check Lambda Calibration* description for more details on the utility and its output).

Click the Master Lamp and than click the `Check Lambda Calibration` button. Look at the log to see the output. In case the result is not satisfactory, re-perform the Create Master Lamp and/or the Adjust First Guess steps.

3.4.8.2 Plot Lambda Calibration

Select a Master Lamp and than click the `Plot Lambda Calibration` button (see the *Plot Lambda Calibration* utility for more details). A new panel will appear. It could be necessary to expand the panel to visualize the plots.

The graphical device has three tabs. In the `One slit` tab, the top panel shows the offset between the real wavelength of the line and that associated by the calibration for all the lines in the *Lines catalog*. Blue points indicate *good* points (see *Create Master Lamp*), red points are those rejected by the fitting procedure (*bad* lines, see *Create Master Lamp*), while magenta triangles are *lost* lines (see *Create Master Lamp*). The latter points are not included in the fit. The `View only good points` button displays only the *good* points in the plot.

The displayed points refers to the `Slit` and `Column` indicated in the Browse panel. The default `Column` is the central one. The user can change slit and position within the slit using the `Previous` and `Next` buttons or scrolling the cursor on the relative bars. Below the top plot a summary of the quality of the fit is reported: the number of *good* lines, the mean value of the offsets and the rms of the distribution, considering both all of the points and only *good* points.

The utility offers the opportunity to exclude *good* points from the fit and repeat the fitting procedure. A point can be excluded by clicking on it with the right button of the mouse. Its color/shape will change from

blue point to red diamond. Click the `Refit` button on the right of the panel. A new fit will be performed, and the mean and rms values will be updated. The excluded point is shown as black triangle. If the new fit is considered worse than the original one, the `Restore the original fit` button restores the plot and numbers to the original status.

In the `One Line` tab, the user can focus his/her attention on a specific line using the menu at the bottom of the panel. Once a line is selected, the deviations measured in all of the columns of the slit for that line is displayed in the top plot as a function of the slit.

Finally, in the `Summary` tab, the distribution of the rms for all slits (estimated for the whole slit at the central position), and a summary table are reported. In this table all the columns can be sorted by clicking on the top.

3.5 Preliminary Reduction

When the Bad pixel Image, the Master Flat and optionally the Master Dark or Master Bias have been created "pre-reduction" of the science frames can start. In this first step the user can clean the frames removing the detector signature and artifacts as bad pixels and cosmic rays. The recipe produces new frames, with a suffix used to know the operation performed by the preliminary reduction (see *Preliminary Reduction*).

To preliminary reduce data, check the parameter values of the `Preliminary Reduction` recipe going into `Parameter Files` menu. If the `CLEAN_BAD_PIX` keyword is activated, append the appropriate bad pixel map to science raw frames. Select the raw frames to polish, the Master Flat (optionally the Master Bias or Master Dark) and then click the `Preliminary Reduction` button.

Caution: Take care to select the Master Flat with the same filter/dichroic/arm as the raw science.

At the end of the process, the pre-reduced polished files (e.g. `BF`, or `BFC`, or `DF`, or `DFC`, see *Preliminary Reduction* for more details) appear in the *Reduction Unit*.

3.6 Reduce Observations

This recipe calibrates in wavelength the 2D spectra, optionally subtracts the sky and calibrates in flux, and extracts the 1D spectra of the detected objects (for more details see *Reduce Observations*).

It requires in input the pre-reduced frames and a Master Lamp. If flux calibration is desired, a *sensitivity function* must be provided as input. In MODS reduction, if user wants to correct spectra for atmospheric extinction an *extinction table* must be provided as input.

To reduce observations, check the parameter values of the `Reduce Observations` recipe going into `Parameter Files` menu.

Select the input pre-reduced frames and the Master Lamp (and optionally the sensitivity function and the extinction table) and click on the `Reduce Observations` button in the `Reduction` tab. A new pop-up window appears. The usage of this window is related - to some extent - to the reduction strategy adopted, particularly to the sky subtraction method.

- **No sky subtraction or sky subtraction with SKY-METHOD** In this case, the order of the input files is not relevant, as well as the presence or not of dithering. The pop-up window directly

summarizes the reduction conditions, and the user can click on the `Run Recipe` button. The recipe will execute all the steps requested by the user.

- **sky subtraction with ABBA-METHOD** In the ABBA-METHOD the sky subtraction is performed subtracting from each frame the subsequent dithered one. In this case, the pop-up window appears with two panels: Science frames and Sky frames. The recipe subtracts from each frame of the Science frames panel the corresponding frame in the Sky frame panel. The couple of frames **must** be dithered. To easily recognize dithered frames, the user can click on the `Get From Header` button in the OFFSETS area. The recipe computes the offsets of input files from their header and updates the Offset column in the Science frames panel. With the help of these offsets, the user can upload the Sky frames files in the right order and then run the recipe. The user can modify the order of the input files in both panels dragging and dropping them. Clearly, despite the name, this method can be used also for observations with dithering pattern more complex then ABBA (e.g. ABAB, ABCDABCD, etc…).
- **sky subtraction with DAVIES-METHOD** Similarly to ABBA-METHOD, the DAVIES-METHOD subtracts from each frame the subsequent dithered one. In this case the pop-up window shows two panels: Science frames and Sky frames. If the user passes only Science frames, the recipe subtracts from each frame of the list its subsequent one. From the last frame it subtracts the previous one. Once files have been uploaded in the Science frames list, and offsets optionally computed, the user can modify the order of the input files dragging and dropping them. When the list is ordered, click the `Run Recipe` button. As alternative, the user can provides also Sky frames. In this case the recipe acts as in the ABBA METHOD.

In each slits, it is possible to add other files clicking the `Add Files` button, and remove a file by clicking on it and then on the `Canc` button. Finally the user can save the list using the `Save List` button and can upload files directly from a list, using the `Load List` button. This option could be particularly useful for the ABBA- and DAVIES-METHOD: if the user wants to test different reduction strategies he/she can upload the list skipping all the organizational steps.

At the end of the process, the reduced files *replace* the pre-reduced ones and an *R* is added to the name (e.g. `BFR`, or `BFCR`, or `DFR`, or `DFCR`). The primary extension of the reduced file is the *pre_reduced_frame.fits*. This allows the user to run multiple time the `Reduce Observations` recipe directly on the reduced frames, without repeating the Preliminary Reduction since the recipe reads the pre-reduced frame from their primary extension.

According to the choices made by the user different extensions will be appended to the reduced file. The 2D extracted spectra are stored in the EXR2D extension of the reduced file, while the 1D spectra in the EXR1D extension. For more details on all the extensions, see the description provided in this *section*.

The user may desire to compare results obtained with different reduction strategies (e.g. different sky subtraction methods). To this aim, SIPGI offers the possibility to change the name to a single file or to a block of files using the context menu (see *Context menus*). The user can select, e.g. the block of reduced files, open the context menu and provide the prefix of the file names. SIPGI renames the selected files using this prefix and enumerates them accordingly to the input frames.

3.6.1 Utilities to check the reduced observations

Once frames are reduced, the user can look at the results and analyze the output using different utilities.

3.6.1.1 Show Reduced Spectra utility.

SIPGI offers the possibility to check the reduced spectra with the *Show Reduced Spectra* utility.

Click on a reduced frame and then click the `Show Reduced Spectra` button in the `Analysis` Tab. A `SPECTRAL VIEWER` window as in Fig. 3.3 will open.

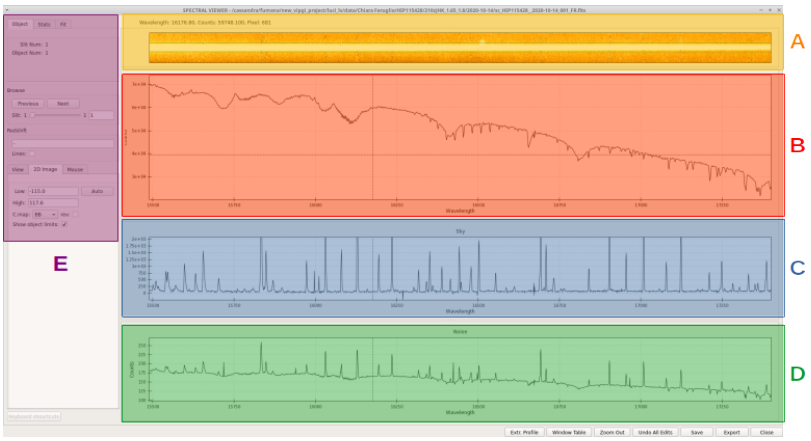

Fig. 3.3: The Show Reduced Spectra layout

In panel A the EXR2D spectra of a selected slit is shown (the selection can be done in panel E, by default slit 1 is shown). The two solid lines indicate the extraction regions of the spectra shown in panel B. Panels C and D show the relative sky and noise spectrum, respectively.

In panel E, the `Object` tab shows the slit number and the number of the object/detection shown in panel B (see *Show Window Table*).

The Browse box allows the user to browse between different slits with the scrolling `Slit` menu. For a given slit, the user can visualize the multiple 1D extracted spectra, if available, with the `Previous` and `Next` button in the same box.

The `View` tab at the bottom of panel E provides the spectrum smoothing tool and the wavelength range selector tool. The `Auto` button shows the EXR2D and 1D spectra in their entirety, i.e. it shows the whole extraction range. The user can change these values according to his/her need. The user can zoom in the 1D (and accordingly in the 2D) spectrum by clicking the left mouse button and dragging it or using the mouse wheel. The `Zoom Out` button at the end of the panel restores the original scale.

The `Smooth` button allows the user to smooth the 1D spectra with a Gaussian kernel with sigma (in pixel) equal to the provided odd number. The `Reset` button restores the spectra to their original shape. To save the smoothed version click the `Save` button at the bottom of the SPECTRAL VIEWER. This will overwrite the original 1D spectra.

The `2D Image` tab allows the user to set the EXR2D display cut levels (i.e. `Low` and `High`) and to modify the 2D image color-map (and revert it). The `Show object limits` flag allows to show or not the extraction limits of the 1D spectra on the EXR2D image (green lines).

Panel E also provides some tools for a quick analysis of the data. The `Mouse` tab permits to perform the following operations with the mouse buttons:

- `STATS`: this option allows to perform quick statistic for a portion of the spectrum. Select the area of interest in the 1D spectrum by clicking the right button of the mouse and dragging it. In the `Stats` tab at the top of Panel E the selected wavelength range, the median of all of the points in the selected wavelength range, the standard deviation (std) of these points and a coarse signal-to-noise ratio (S/N) are displayed. The S/N is computed as the ratio between the median value and the std.
- `EDIT`: this option allows to edit the spectrum. There are two ways to edit spectrum with mouse: *i*) replace a spectrum region with a straight line. Click the right button of the mouse and drag the mouse to select the editing range: a straight line is drawn from the leftmost selected point up to the rightmost selected point. *ii*) replace a punctual spectrum value. Click with the middle button of the mouse a point in the 1D spectrum panel. The task will connect the selected point to spectrum. All edits performed to the 1D spectrum can be undone pressing "u" key on the keyboard (up to the complete restoration of the original spectrum), provided you did not move to a different spectrum. In case something goes really wrong it is also possible to restore the original data and restart the editing from scratch clicking on the button "Undo Edits".
- `FIT`: this option allows to fit spectral lines. Spectral lines can be fitted using two methods: *i*) Gaussian. A Gaussian is fitted in the selected wavelength range (identified clicking and dragging the mouse) and the results are shown in the `Fit` tab at the top of panel E. *ii*) Peak. The peak of the selected line is computed finding the barycenter of the selected points. The results are shown in the `Fit` tab. To clear the fit result from the plot, press "c" key.

In the `Redshift` box, the user can quickly check the source redshift. Putting the expected redshift, and flagging the `Lines` entry, the SPECTRAL VIEWER shows on the 1D spectrum the position of the main absorption and emission lines according with the provided redshift.

`Ext. Profile` button shows the extraction profile (see *Plot Extraction Profile*); `Window Table` button shows the WIN table (see *Show Window Table*). `Export` buttons export the shown 1D, 2D, noise e sky spectrum. Clicking the button, a window appear to provide the name and location of the saved files. If the output file has a FITS extension, the data are saved in FITS file, otherwise in ASCII. `Export All` export *all* the data of the reduced file.

3.6.1.2 Manage Detections utility

The *Manage Detections* utility allows the user to modify/add/delete the objects detections.

Click on a reduced frame and then click the `Manage Detections` button in the `Analysis` Tab. A new window appears, with on the top the EXR2D spectra of the first slit. The user can browse between the slits with the two `Previous` and `Next` buttons at the bottom of the panel.

In the left panel is shown the *extraction profile* of the slit. Passing with the mouse on the plot, a blue line appears both on the profile and on the corresponding point in the EXR2D spectra. Clicking the right mouse button and dragging it, the user can select a new region of the 2D spectra to extract. This region is highlighted with a yellow area. Releasing the right mouse button, the utility automatically adds the detection in the table on the right.

In the right table is shown a summary of all the objects that have been detected in that slit. In the table, the columns *Slit*, *Obj*, *Obj Start*, and *Obj End* indicate the number of the slit which is shown, the number of the detection, the value of pixels at which the object begins and ends. Clicking with the left button of the mouse on a detection, the detection limits are shown on the extraction profile on the left. Clicking with the right button of the mouse on a detection in the table, the user can delete the selected detection clicking on `Delete objects`.

Once the user is satisfied with all the changes, she/he *must* commit the changes clicking the `Commit Changes` button, and only after that he/she can move to the next slit without losing all the work. Once

the user has finished all the changes in all the slits, she/he *must* apply the changes clicking the `Apply Changes` button. At this point the utility extracts all the new detections, removes the deleted ones, and updated the input file accordingly. By default, the extraction is performed with the sum. If the input spectra is in ADU, the user can activate the Horne extraction clicking the `Use Horne Extraction` flag **before** clicking the `Apply Changes` button.

3.6.1.3 Plot Extraction Profile utility

Click on a reduced frame and then click the `Plot Extraction Profile` button in the `Analysis` Tab. It shows the Extraction Profile (see *Plot Extraction Profile*). The `Previous` and `Next` buttons allow to browse between slits/objects.

3.6.1.4 Show Window Table utility

Click on a reduced frame and then click the `Show Window Table` button in the `Analysis` tab. A new window will open showing the WIN table of the selected spectrum (see *Show Window Table*).

3.6.2 Molecfit

SIPGI offers the opportunity to export reduced data outside the SIPGI environment to *treat them with Molecfit*.

Select the files to be exported and then click on the `SIPGI To Molecfit` button in the `Reduction` Tab. A new windows appears with the list of the selected files. The user must provide the recipe with the slit/object number of the detection she/he wants to export filling in the two relative columns in the panel. To identify these numbers the user can use the *Show Reduced Spectra* utility.

Click the `Run` button and provide the directory in which to save the files.

Once the transmission function has been calculated with Molecfit, the user can apply it to data *in* SIPGI. This allows the user to continue to work in SIPGI with corrected frames. To do this, select the files to be corrected, click the `Apply Molecfit Tra` button in the `Reduction` Tab, and a new window will appear with two panels. In the left panel, the list of the files to be corrected is shown. The user must load in the right panel the transmission functions derived with Molecfit. If just one function is provided, it will be applied to all the files in the left panel. The `Run` button executes the recipe.

SIPGI offers the possibility to revert this operation with the `Revert Molecfit Tra` button. Select the files to revert to their original form and click the `Revert Molecfit Tra` button.

3.7 Sensitivity function

To account for the instrument response and to covert spectra in physical unit, it is necessary to *create a sensitivity function*.

3.7.1 MODS

In MODS observations the recipe requires the wavelength-calibrated spectra of the standard star (e.g. BFCR frames) and its *Spectro-photometric standard*. The recipe compares 1D reduced spectra of the observed standard star and its Spectro-photometric standard obtaining a raw sensitivity function. If the Spectro-photometric standard does not cover the whole wavelength range, the recipe accepts in input a template from the provided Pickle's library to extend the coverage (see *create sensitivity*).

Before starting the computation of the sensitivity function it is useful compare the observed data with the Spectro-photometric standard. To do this, click the standard BFCR frame *and* the Spectro-photometric standard and then click the `Show Reduced Spectra` button in the `Analysis` tab. The Spectro-photometric standard is displayed on top of the 1D spectra (in panel B of Fig. 3.3). This visualization helps both into understanding the extension of the Spectro-photometric standard (and hence if it is necessary to provide a Pickle template in input) and also its quality.

To obtain the sensitivity function for MODS observations, check the parameter values of the `Create Sensitivity` recipe going into `Parameter Files` menu.

Select the wavelength-calibrated frames of the standard stars, its Spectro-photometric standard (optionally, a Pickle template of the same spectral type/class of the standard star from the *Generic Master Frames and Calibration Files*) and click the `Create Sensitivity` button in the `Reduction` Tab. A new panel opens with the list of input frames. The recipe identifies the standard star as the detection with the highest flux in each frame. It could happen that the detection with the highest flux is not the standard star. In this case the user must provide the recipe with the slit/object number of the standard star detection filling in the two relative columns in the panel. To identify these numbers the user can use the *Show Reduced Spectra* utility. Click the `Run` button and the sensitivity function will be created.

3.7.2 LUCI

In LUCI observations, the recipe requires the wavelength-calibrated frames of the telluric. Matching the information stored in the telluric FITS header and those in the *Hipparcos catalog*, the recipe identifies the spectral type of the observed telluric and automatically selects the relative *Stellar template*. If the header keywords are not correctly filled, the user must also provide in input a Stellar template of the same spectral type of the observed telluric. The recipe re-scales the Stellar template to the nominal magnitude of the star in a given filter and computes the sensitivity function.

To obtain the sensitivity function for LUCI observations, check the parameter values of the `Create Sensitivity` recipe going into `Parameter Files`.

Select the wavelength-calibrated frames of the telluric. If the telluric is not in the Hipparcos catalog, provide a template of the same spectra type/class of the telluric. Click the `Create Sensitivity` button in the `Reduction` Tab and then follow the instruction for MODS Sensitivity function.

Important: For MOS observation, see the *MOS LUCI sensitivity function* paragraph.

3.7.3 Utility to check the sensitivity function: Plot Sensitivity

Once the sensitivity function is derived, the user can look at the result using the *Plot Sensitivity utility*. Select the sensitivity function and then click the `Plot Sensitivity` button in the `Analysis` tab. A new window with two panels will appear.

In the bottom panel, the red crosses show the raw sensitivity function, i.e. the function obtained by dividing the median wavelength calibrated observed spectrum by its reference model. The blue line is the smoothed (final) sensitivity function.

In the top panel:

- for LUCI observations, the magenta solid line is the Stellar template scaled to the telluric magnitude used to derive the sensitivity. Black crosses are the flux-calibrated median spectrum of input telluric spectra.
- for MODS observations, the magenta solid line is the Spectro-photometric standard used to derive the sensitivity. Black crosses are the flux-calibrated median spectrum of input standard spectra.

In each panel, the user can zoom the functions clicking the left mouse button and dragging it or using the mouse wheel.

The utility allows to also check the sensitivity function on the single telluric/standard input spectra.

The user has to re-reduce the standard/telluric frames applying the flux calibration. Once these files have been obtained, select the sensitivity function and then click the `Plot Sensitivity` button in the `Analysis` tab.

In the bottom part of the window the `Plotted Frame` menu allows the user to select the spectrum shown in the upper panel; the median used to compute the sensitivity itself (default value) or one of the flux-calibrated star (if selected as input).

The user can also modify the final sensitivity function by clicking the `Edit mode` flag. This opens many possibilities: the user can *i)* smooth the sensitivity with a mean filter with dimension `Smooth level` pixel; *ii)* edit the sensitivity pixel by pixel by clicking with the central button of the mouse; *iii)* replace the sensitivity with a straight line in the region highlighted by clicking the mouse right button and dragging it.

Once the user is satisfied with the changes, he/she **must** verify the new sensitivity on data clicking the `Reapply Sensitivity` button. This will upgrade the flux-calibrated spectrum in the top panel according with the new sensitivity. If the user is satisfied with that, he/she can save the modified version with the `Save` button, otherwise the `Undo All` button resets all of the changes.

3.8 Combine Observations

The *combine observation recipe* takes in input the `DFCR`, `BFCR` frames (flux calibrated or not) and stack them together to increase the signal-to-noise ratio. Then, for each slit, it determines the extraction profiles and extracts the 1D spectra of the detected objects.

The output file is a brand new file, which will contain several extensions.

To combine frames, check the parameter values of the `Combine Observations` recipe going into `Parameter Files`.

Select all of the frames to combine and click the `Combine Observations` button in the `Reduction` tab. This opens a new panel with the input frames listed plus a column with the offsets.

The recipe offers different options to compute the offsets: `Get From Headers`, `Compute From WIN` and `Compute from EXR2D` (see *Combine observations* for more details on the methods). Clicking the `Get From Headers` button, the recipe is directly launched. On the contrary, the `Compute From WIN` and `Compute from EXR2D` buttons opens the relative `Parameter` panels of the recipes. The user can set there their customized parameters and then launch the recipe clicking the `Compute` button.

The user can also edit the Offset column and insert offsets values by hand.

As for the `Reduce Observations`, the recipe offers the opportunity to save and upload a list of files with the `Save List` and `Load List` button.

The combined frame can be visualized and checked with the `Show Reduced Spectra`.

CHAPTER

FOUR

AUXILIARY SYSTEM FILES

Apart from the raw data (e.g. bias, dark, flat, lamp, scientific frames) acquired during the observing runs, the reduction process requires in input Auxiliary system files.

SIPGI is released with a set of Auxiliary system files necessary to the reduction. These auxiliary files can be grouped in two main classes: the instrument files and the calibration files.

Important: Together with the DRS, the Auxiliary system files are the **core of SIPGI**; these files are used to properly calibrate and extract spectra from frames acquired with different instrument configurations.

Both instrument and calibration files are appended to raw files by the built-in data organizer during the import.

4.1 Instrument files

The instrument files are of three types:

1. *Paf files*
2. *Grism tables*
3. *CCD tables*

4.1.1 Paf files

The *Paf files* are ASCII files that depend on the instrument configuration and on the mask type, long slit (LS) or multi-object spectroscopy (MOS). They contain the coefficients of the models used by SIPGI to locate and wavelength calibrate 2D spectra. These geometrical models are:

4.1.1.1 The Optical Model

The Optical (OPT) Model assigns the mask slit position (provided by the mask constructor in millimeters) to the position (in pixels) of the grism central wavelength ($\lambda_{\rm ref}$, see *grism table*) on the detector. Hereafter this position will be referred as slit reference position. The model takes into account the known optical distortions of the instrument. The OPT model does not depend on the grism in use, since it just locates the slit position on the detector ignoring any information about dispersion.

The model is defined by a pair of global 2D polynomials of the same degree: X describing distortions along the x axis (i.e. along the cross dispersion direction) and Y along the y axis (i.e. along the dispersion

direction):

$$x_{[pix]} = \sum_{i=0}^{2} \sum_{j=0}^{2} X_{i,j} x_{[mm]}^{i} y_{[mm]}^{j}$$

$$y_{[pix]} = \sum_{i=0}^{2} \sum_{j=0}^{2} Y_{i,j} x_{[mm]}^{i} y_{[mm]}^{j}$$

The *Paf file* contains the set of $X_{i,j}$ and $Y_{i,j}$ coefficients.

4.1.1.2 The Curvature Model

The Curvature (CRV) Model describes the 2D spectra displacement with respect to the ideal dispersion direction, perfectly aligned along the pixels grid. Due to optical distortions on real data, spectra are not perfectly aligned along the pixels and these distortions change within the FOV. The CRV Model describes these deviations (in pixel) of the spectral trail wrt the theoretical dispersion direction.

For each slit, one mono dimensional polynomial (of order N) is used to describe the displacement Δc along the cross dispersion direction, starting from the slit reference position (located by the OPT Model) (see Fig. 4.1)

$$\Delta c_{[pix]} = \sum_{i=0}^{N} a_{x,y,i} \Delta d_{[pix]}^{i}$$

Δd is the displacement with respect to the slit reference position in pixel.

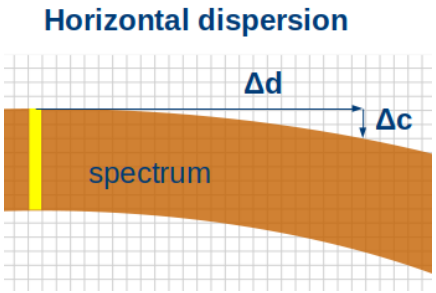

Fig. 4.1: The figure shows the distorted 2D spectrum (orange area) wrt the ideal horizontal dispersion direction on the detector. The yellow region indicates the position of the slit. The CRV Model estimates the amount of distortion Δc as a function of Δd, i.e. the distance from the slit.

Each slit has its own set of $a_{x,y,i}$ coefficients because each slit is in a different position in the FOV.

SIPGI uses a global model (the CRV Model) to describe the coefficients variation across the FOV. In particular, the local $a_{x,y,i}$ coefficients depend on the computed global model (A) at the slit reference position on detector:

$$a_{x,y,i} = \sum_{h=0}^{N} \sum_{k=0}^{M} A_{i,h,k} x_{[pix]}^{h} y_{[pix]}^{k},$$

where N and M are the order of x and y polynomial, respectively. The *Paf file* contains the set of $A_{i,h,k}$ coefficients.

4.1.1.3 The Inverse Dispersion Solution Model

The Inverse Dispersion Solution (IDS) Model describes the wavelength to pixel relation. Once the slit positions are identified on the detector (by the OPT Model) and the tracing of the 2D dispersed spectra is reconstructed (by the CRV Model) the wavelength calibration of the 2D spectra can be carried out. The IDS Model moves along the tracing curves described by the combination of the OPT and CRV Models and it assigns the expected wavelength values to each pixel along this curve.

The IDS Model mathematical description is quite similar to the CRV Model: for each slit, one mono dimensional polynomial (of order N) locates the wavelength position wrt λ_{ref}:

$$\Delta d_{[pix]} = \sum_{i=0}^{N} b_{x,y,i}(\lambda - \lambda_{ref})^i,$$

where Δd is the displacement of λ in pixels with respect to λ_{ref}.

Since each slit is in a different position in the FOV, and distortions change within the FOV, each slit has its own set of $b_{x,y,i}$ coefficients. A global 2D polynomial (D) is used to describe the variation of $d_{x,y,i}$ across the FOV:

$$b_{x,y,i} = \sum_{h=0}^{N}\sum_{k=0}^{M} B_{i,h,k} x_{[pix]}^h y_{[pix]}^k,$$

where N and M are the order of x and y polynomial, respectively. The *Paf file* contains the set of $B_{i,h,k}$ coefficients.

Note: For a given instrumental configuration, the OPT, CRV and IDS Models clearly differ whether they have to describe LS or MOS observations. Despite this, *if the instrument is stable*, once the three Models have been calibrated on a set of LS(MOS) data, the same Models can be used to describe every LS(MOS) set of data acquired with the same configuration.

SIPGI comes with a set of default *Paf files* (see table below) covering the standard MODS and LUCI instrument configurations, both LS and MOS. For LUCI observations, the provided *Paf files* refer to the nominal central wavelength of the configuration (i.e. λ_{ref} in the formula above) unless otherwise specified. If data were acquired with a central wavelength different from the nominal one, the user can "adjust" the *Paf file* as described in *Adjust First Guesses* paragraph.

Table 4.1: The *Paf files* released with SIPGI. Column 1: Instrument; Column 2: Grism; Column 3: Dichroic; Column 4: λ_{ref}; Column 5: Camera; Column 6: Mask type; Column 7: Paf file name. The "#" could be 1 or 2 to indicate files for the two different LBT arms.

Instrument	Grism	Dichroic	λ_{ref}	Camera	Mask	Paf file
MODSR	G670L				LS	MODS#R_LS_G670L_Red.paf
MODSR	G670L				MOS	MODS#R_MOS_G670L_Red.paf
MODSR	G670L_dual	Dual			LS	MODS#R_LS_G670L_Dual.paf
MODSR	G670L_dual	Dual			MOS	MODS#R_MOS_G670L_Dual.paf
MODSB	G400L				LS	MODS#B_LS_G400L_Blue.paf
MODSB	G400L				MOS	MODS#B_MOS_G400L_Blue.paf
MODSB	G400L_dual	Dual			LS	MODS#B_LS_G400L_Dual.paf
MODSB	G400L_dual	Dual			MOS	MODS#B_MOS_G400L_Dual.paf
LUCI	150Ks		2.17 μm[1]	N1.8	LS	LUCI#_LS_150Ks_2.15_1.8.paf
LUCI	150Ks		2.17 μm[1]	N1.8	MOS	**NO DATA AVAILABLE**
LUCI	200H+K		1.17 μm	N1.8	LS	LUCI#_LS_200H+K_1.17_1.8.paf
LUCI	200H+K		1.17 μm	N1.8	MOS	LUCI#_MOS_200H+K_1.17_1.8.paf
LUCI	200H+K		1.93 μm	N1.8	LS	LUCI#_LS_200H+K_1.93_1.8.paf
LUCI	200H+K		1.93 μm	N1.8	MOS	LUCI#_MOS_200H+K_1.93_1.8.paf
LUCI	210zJHK		0.95 μm[2]	N1.8	LS	LUCI#_LS_210zJHK_0.95_1.8.paf
LUCI	210zJHK		0.95 μm[2]	N1.8	MOS	**NO DATA AVAILABLE**
LUCI	210zJHK		1.25 μm	N1.8	LS	LUCI#_LS_210zJHK_1.25_1.8.paf
LUCI	210zJHK		1.25 μm	N1.8	MOS	LUCI#_MOS_210zJHK_1.25_1.8.paf
LUCI	210zJHK		1.65 μm	N1.8	LS	LUCI#_LS_210zJHK_1.65_1.8.paf
LUCI	210zJHK		1.65 μm	N1.8	MOS	LUCI#_MOS_210zJHK_1.65_1.8.paf
LUCI	210zJHK		2.20 μm	N1.8	LS	LUCI#_LS_210zJHK_2.20_1.8.paf
LUCI	210zJHK		2.20 μm	N1.8	MOS	LUCI#_MOS_210zJHK_2.20_1.8.paf

4.1.2 Grism tables

The *Grism tables* are FITS files that depend on the instrument configuration. They supply SIPGI with:

- the reference lambda in Angstrom (λ_{ref}), i.e. the central wavelength of the configuration (*WLEN_CEN* keyword); through the *Paf file* SIPGI associates the slit position on the frame with the λ_{ref} position;
- the tracing limits (in pixels) wrt the slit position (*LLEN_LO* and *LLEN_HI* keywords). Starting from the location of the slit position on the detector, the tracing of the 2D spectra will be performed in the range *[slit_position - LLEN_LO, slit_position + LLEN_HI]*;

[1] Nominal wavelength 2.15
[2] Nominal wavelength 0.97

- the 2D spectra extraction limits: *WLEN_START*, *WLEN_END* keywords (in Angstrom);
- the spectral sampling of 2D and 1D spectra: *WLEN_INC* keyword (in Angstrom).

Note: The spectral tracing is performed in the range *[slit_position-LLEN_LO,slit_position+LLEN_HI](pixels)*. The *Grism tables* provided with SIPGI are such that extraction range *[WLEN_START, WLEN_END]* () is included in the *[slit_position-LLEN_LO,slit_position+LLEN_HI]* range (pixels) according to the first guesses solution contained into the *Paf file*.

SIPGI supplies the user with a set of *Grism tables* covering all the standard instrument configurations. The list of these *Grism tables* with their default extraction limits and spectral sampling is reported below. The extraction limits reported in the LUCI *Grism tables* are tuned on real data and on the nominal central wavelength (see λ_{ref} in table below) of the specific configuration. For MODS data, the extraction limits correspond to those reported in the MODS webpage. The default sampling is the nominal dispersion of the configuration.

Table 4.2: The *Grism Tables* released with SIPGI. Column 1: Instrument; Column 2: Grism; Column 3: Dichroic; Column 4: λ_{ref}; Column 5: Camera; Column 6: Grism table name; Column 7: Extraction limits in Angstrom; Column 8: Spectral sampling.

Instrument	Grism	Dichroic	λ_{ref}	Camera	Grism Table file	Extraction limits	Spectral sampling
MODSR	G670L				grismTable_G670L.fits	5000-10000	0.85 /px
MODSR	G670L_dual	Dual			grismTable_G670L_Dual.fits	5400-10000	0.85 /px
MODSB	G400L				grismTable_G400L.fits	3100-6000	0.52 /px
MODSB	G400L_dual	Dual			grismTable_G400L_Dual.fits	3200-5900	0.52 /px
LUCI	150Ks		2.17 μm	N1.8	grismTable_150Ks_2.17_N1.8.fits	19800-23200	2.59 /px
LUCI	200H+K		1.17 μm	N1.8	grismTable_200H+K_1.17_N1.8.fits	9600-13680	2.16 /px
LUCI	200H+K		1.93 μm	N1.8	grismTable_200H+K_1.93_N1.8.fits	15180-23300	4.32 /px
LUCI	210zJHK		0.95 μm	N1.8	grismTable_210zJHK_0.95_N1.8.fits	8330-10060	0.64 /px
LUCI	210zJHK		1.25 μm	N1.8	grismTable_210zJHK_1.25_N1.8.fits	11745-13170	0.76 /px
LUCI	210zJHK		1.65 μm	N1.8	grismTable_210zJHK_1.65_N1.8.fits	15490-17400	1.01 /px
LUCI	210zJHK		2.20 μm	N1.8	grismTable_210zJHK_2.20_N1.8.fits	20400-23600	1.63 /px

If the user is not satisfied with these values, he/she can modify them during the reduction (see *Adjust First Guesses* and *Reduce Observations*). The *Grism table* includes also the wavelength in Angstrom of a set of sky lines covered by the spectral configuration. These lines will be used to check and refine directly on scientific data the wavelength calibration obtained from lamp frames (see *Reduce Observations*).

During the import of raw data, the built-in data organizer retrieves from the FITS header the information on the instrument configuration and append to each file the suitable *Paf file* and *Grism table*. For LUCI observations with central wavelength different from the nominal one, SIPGI appends the files related to the nominal configuration. As said before, if the instrument is stable and central wavelength is the nominal one, SIPGI can "automatically" locate the spectra and wavelength calibrate them using the information stored in the *Paf files*. Nonetheless, instruments are not perfect, optical distortions can change on a night base for several reasons. Furthermore masks could be not perfectly aligned in the FPU or the central wavelength might not be the nominal one. For these reasons, SIPGI offers the users the possibility to check and refine the default *Paf file* on real data (see *Adjust First Guesses*).

4.1.3 CCD tables

The *CCD tables* are FITS files containing information related to the CCD such as gain and readout noise organized by quadrants. The *CCD table* will also store the list of bad pixels on the detector if provided by the user (pixels enumeration starts from 1) (see *Append Bad Pixel Image*).

SIPGI comes with the following *CCD tables*:

Instrument	Readout mode	CCD Table file
MODS#R		MODS#R_ccdTable.fits[3]
MODS#B		MODS#B_ccdTable.fits[3]
LUCI#	LIR	LUCI#_LIR_ccdTable.fits[4]
LUCI#	MER	LUCI#_MER_ccdTable.fits[4]
LUCI1	DCR	LUCI1_HAWAII_DCR_ccdTable.fits[5]
LUCI1	MER	LUCI1_HAWAII_MER_ccdTable.fits[5]

During the ingestion of raw frames, the built-in data organizer appends also the *CCD table* to the raw files, together with *Paf file* and *Grism table*.

4.2 Calibration files

The calibration files are those necessary for calibration purposes. SIPGI has two types of calibration files, those necessary for the creation of the sensitivity function (*Spectro-photometric standard*, *Stellar templates*) and those necessary for the wavelength calibration (*Lines catalogs*).

Note: In this SIPGI version, MODS and LUCI calibration files have wavelengths **in vacuum**. In case the original files were defined in air wavelengths (see e.g. *Stellar template*), the conversion from air to vacuum wavelength has been performed by the PyAstronomy library using the Edlen (1953) and Ciddor (1996) prescriptions. All the MODS spectra released by LBT spectroscopic data reduction center have wavelengths in air.

4.2.1 Spectro-photometric standard

The *Spectro-photometric standards* are ASCII files describing the spectrum of standard stars. SIPGI comes with a set of *Spectro-photometric standard* files listed in *Appendix A*. For MODS observations, the sensitivity function is derived by comparing the wavelength-calibrated 1D spectra of the observed standard star with the corresponding *Spectro-photometric standard* (for more details see *Create Sensitivity*).

Caution: Users are strongly encouraged to check the resolution of the provided *Spectro-photometric standards* and to verify they are suitable for their scientific purposes (see *Appendix A*). If this is not the

[3] see MODS LBT pages for more details

[4] see LUCI LBT pages for more details.

[5] These *CCD tables* are provided for old data acquired with HAWAII-2 HdCdTe detector. See old LUCI manual (Table 8) for more details.

case, user can upload in SIPGI a *customized Spectro-photometric standard* file through the `Custom Auxiliary Files` *panel*.

4.2.2 Stellar templates

The *Stellar templates* are ASCII files with the theoretical spectra of stars of different type and class. SIPGI is released with a series of Pickles (Pickles, 1998) *Stellar templates* listed in *Appendix A*. For LUCI observations, the sensitivity function is derived comparing the wavelength-calibrated 1D spectra of the observed telluric with the theoretical template of a star of the same spectral type and class. Before comparing real data with the template, SIPGI scales the last one to nominal fluxes/magnitude of the observed telluric. The nominal magnitude for a significant sample of tellurics are listed in the SIPGI Hipparcos Catalog.

The SIPGI Hipparcos Catalog is provided by SIPGI and contains the spectral type/class and the nominal 2MASS-J, -H, -K magnitude for a sample of 112404 Hipparcos stars (Zacharias et al. 2004).

Caution: Users are strongly encouraged to check the resolution of the *Stellar templates* provided and to verify they are suitable for their scientific purposes (see *Appendix A*). If this is not the case, user can upload in SIPGI a *customized Stellar template* through the `Custom Auxiliary Files` *panel*.

4.2.3 Lines catalogs

The *Lines catalogs* are FITS files containing the wavelengths (in Angstrom) of the main bright isolated emission lines of calibration lamps. These catalogs must cover the whole spectral range of interest in order to provide the most accurate wavelength calibration of the spectra. For this reason the content of these catalogs depends on arc lamps, grism and camera. These files are appended to lamp frames by the built-in data organizer.

Starting from the theoretical *IDS Model* stored in the *Paf file*, SIPGI is able to predict and show the position of the *Lines catalogs* lines on the lamp frames (see *Adjust First Guesses*). This will allow the user to visually check and refine the lambda-pixel relation and hence to perform the wavelength calibration in an easier and quicker way. Together with the lamp *Lines catalogs*, SIPGI has also sky *Lines catalogs* that list the wavelengths of the brightest sky emission lines, to be used if the wavelength calibration has to be performed directly on scientific frames. SIPGI comes with the following *Lines catalogs*:

Table 4.3: The *Lines catalogs* released with SIPGI. Column 1: Instrument; Column 2: Grism; Column 3: Camera; Column 4: arc element(s); Column 5: Lines catalog name.

Instrument	Grism	Camera	Lines	Line Catalog
MODS	G400L/G670L		Ar	lineCat_Ar.fits
MODS	G400L/G670L		Hg	lineCat_Hg.fit
MODS	G400L/G670L		Kr	lineCat_Kr.fits
MODS	G400L/G670L		Ne	lineCat_Ne.fits
MODS	G400L/G670L		Xe	lineCat_Xe.fits
MODS	G400L/G670L		Ar Ne	lineCat_ArNe.fits
MODS	G400L/G670L		Hg Ne	lineCat_HgNe.fits

continues on next page

Table 4.3 – continued from previous page

Instrument	Grism	Camera	Lines	Line Catalog
MODS	G400L/G670L		Kr Xe	lineCat_KrXe.fits
MODS	G400L/G670L		Ar Hg Ne	lineCat_ArHgNe.fits
MODS	G400L/G670L		Ar Kr Xe	lineCat_ArKrXe.fits
MODS	G400L/G670L		Hg Kr Xe	lineCat_ArHgKrXe.fits
MODS	G400L/G670L		Ar Hg Kr Xe	lineCat_ArHgKrXe.fits
MODS	G400L/G670L		Ar Kr Ne Xe	lineCat_ArKrNeXe.fits
MODS	G400L/G670L		Hg Kr Ne Xe	lineCat_HgKrNeXe.fits
MODS	G400L/G670L		Ar Hg Kr Ne Xe	lineCat_ArHgKrNeXe.fits
MODS	G400L/G670L		Sky lines	lineCat_Sky.fits
LUCI	150Ks	1.8	Ar	lineCat_Ar_150Ks_1.8.fits
LUCI	150Ks	1.8	Ne	lineCat_Ne_150Ks_1.8.fits
LUCI	150Ks	1.8	Ar Ne	lineCat_ArNe_200H+K_1.8.fits
LUCI	150Ks	1.8	Sky lines	lineCat_sky_150Ks_1.8.fits
LUCI	200HK	1.8	Ar	lineCat_Ar_200H+K_1.8.fits
LUCI	200HK	1.8	Ne	lineCat_Ne_200H+K_1.8.fits
LUCI	200HK	1.8	Xe	lineCat_Xe_200H+K_1.8.fits
LUCI	200HK	1.8	Ar Xe	lineCat_ArXe_200H+K_1.8.fits
LUCI	200HK	1.8	Ne Xe	lineCat_NeXe_200H+K_1.8.fits
LUCI	200HK	1.8	Ar Ne Xe	lineCat_ArNeXe_200H+K_1.8.fits
LUCI	200HK	1.8	Sky lines	lineCat_sky_200H+K_1.8.fits
LUCI	210zJHK	1.8	Ar	lineCat_Ar_210zJHK_1.8.fits
LUCI	210zJHK	1.8	Ne	lineCat_Ne_210zJHK_1.8.fits
LUCI	210zJHK	1.8	Xe	lineCat_Xe_210zJHK_1.8.fits
LUCI	210zJHK	1.8	Ar Ne	lineCat_ArNe_210zJHK_1.8.fits
LUCI	210zJHK	1.8	Ar Xe	lineCat_ArXe_210zJHK_1.8.fits
LUCI	210zJHK	1.8	Ne Xe	lineCat_NeXe_210zJHK_1.8.fits
LUCI	210zJHK	1.8	Ar Ne Xe	lineCat_ArNeXe_210zJHK_1.8.fits
LUCI	210zJHK	1.8	Sky lines	lineCat_sky_210zJHK_1.8.fits

SIPGI *Lines catalogs* are obtained using these references:

- Argon lines from IRAF (see argon.dat);
- Xenon lines from IRAF (see xenon.dat);
- The NIST data base;
- The HO lines atlas provided by: Rousselot et al. (2000); lowd_ir_ohlines.dat; ir_ohlines.dat; Osterbrock et al. (1996)

4.3 Mask files

The *Mask file* is the file that describes the mask for MODS data. The description of the mask is necessary to SIPGI to locate the slit(s) position on the CCD. LUCI does not require these kind of files since this information is stored in the FITS header of raw frames. The *Mask files* for standard MODS long-slit (LS) observations are released with SIPGI. SIPGI comes with these LS *Mask files*:

Mask file	Slits Number	Slit aperture
LS5x60x0.3.mms	5	0.3 arcsec
LS5x60x0.6.mms	5	0.6 arcsec
LS5x60x0.8.mms	5	0.8 arcsec
LS5x60x1.0.mms	5	1.0 arcsec
LS5x60x1.2.mms	5	1.2 arcsec
LS60x1.0.mms	1	1.0 arcsec
LS60x5.mms	1	5.0 arcsec

For MODS MOS observations, the user **must** supply SIPGI with the *Mask file* of its observations (see the *Add mask definition files*). The mask to upload is the file produced by the LBT mask preparation software.

4.4 Extinction table

The *Extinction Table* is an ASCII file containing the model extinction curve for LBT. It is provided by LBT (see MODS page).

CHAPTER

FIVE

DATA ORGANIZATION

The built-in data-organizer organizes files within a hierarchical scheme composed of 4 classes:

- *Workspace*
- *Project*
- *Dataset*
- *Reduction Unit*

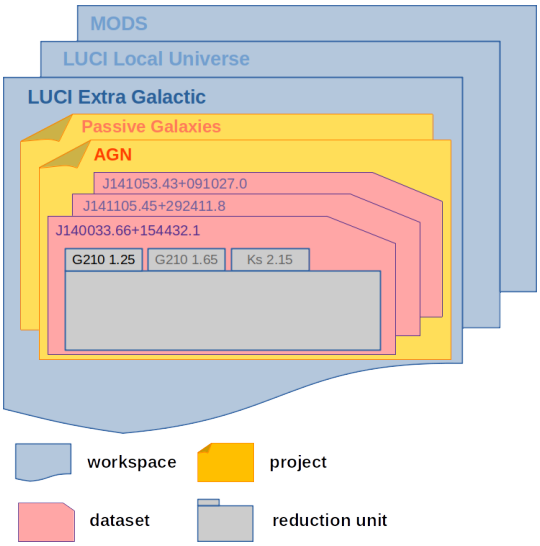

Fig. 5.1: This figure shows the SIPGI data organization scheme.

5.1 Workspace

A *Workspace* is a SIPGI environment that depends on the instrument (e.g. MODS or LUCI) and **must** be defined by the user before starting the ingestion of raw frames. In a *Workspace* all the recipes necessary to reduce and calibrate data of the selected instrument are activated. The behavior of each recipe depends on a number of *parameters*. When a *Workspace* is created, the parameters of all the recipes are set to their default values. If a user changes the parameters of a specific recipe, these changes are valid throughout all of the *Workspace*.

5.2 Project

A *Project* is the ensemble of all the data necessary to the reduction (both science and calibration frames). Several *Projects* may exist within the same *Workspace*. Before ingesting raw data, the user **must** specify both the *Project* name and a Project directory on his/her disk in which the built-in data organizer will store a copy of the raw frames in the SIPGI format (see *The importing procedure*).

Users can create many *Workspaces* and *Projects*. This allows to organize reductions in the most efficient way. For example, the user can create a *Workspace* with all the setting parameters tuned for the reduction of some targets and another *Workspace* with different settings best suited to other targets.

5.3 Dataset

A *Dataset* is the collection of data with the same characteristics (see later). *Datasets* are defined by SIPGI. The organizer splits all of the raw data in several *Datasets* within a *Project*. *Datasets* are of two types: scientific datasets and calibration datasets. Scientific datasets contain data belonging to the same PI and of the same target. Normally, for one run, the user will have a scientific dataset collecting the frames of the standard star and as many scientific datasets as the number of targets he/she observed. Each dataset collects the frames of one target. Calibration datasets contain calibration data (flat fields, lamps, darks, bias, etc…). The "organizer" groups calibration frames in different *Datasets* depending on whether they are imaging frames, LS frames or MOS frames.

The type of file (e.g. science or calibration), the PI and target name, the observing mode (e.g. imaging, LS, MOS) are all identified by the organizer by reading and combining specific keywords in the header of the raw files. These keywords are automatically written in the header of the file at the time of the observations. SIPGI will not digest and categorize raw files with incorrect entries for these specific keywords. In *Appendix B* the list of keywords required by SIPGI for a successful import are reported.

5.4 Reduction Unit

In each *Dataset*, SIPGI organizes raw files in *Reduction Units*. A *Reduction Unit* collects all the files with the same instrument configuration, i.e. the same instrument, grism, filter, and mask. As example, the scientific frames of a target observed with MODS with the G400L and the G670L grisms will be split in two *Reduction Units*: one for the G400L data on both instrument arms, and one for the G670L on both arms[1]. Within each *Reduction Unit*, data can be further grouped together according to, e.g., the observing night, the exposure time, etc… using filter facilities provided by SIPGI (see *Frames panel*). This organization allows the user to easily have at disposal the group of data that have to be processed together minimizing the possibility that the wrong input data is provided to the SIPGI reduction recipes.

[1] SIPGI considers left and right arm as the same instrument and hence collects data of both arms in the same *Reduction Unit*. This is done to allow the stacking of frames acquired on both arms.

5.5 The importing procedure

During the import of the data in their *Datasets* and *Reduction Units*, SIPGI:

- **Rotates and flips the files** The orientation of raw data is modified. This is mostly for historically reasons. In particular, raw LBT frames are rotated by 90 degrees and flipped. This means that while in raw frames the spectral dispersion is along the x-axis, in SIPGI raw spectra are dispersed along the y-axis, with wavelength increasing from bottom to top.

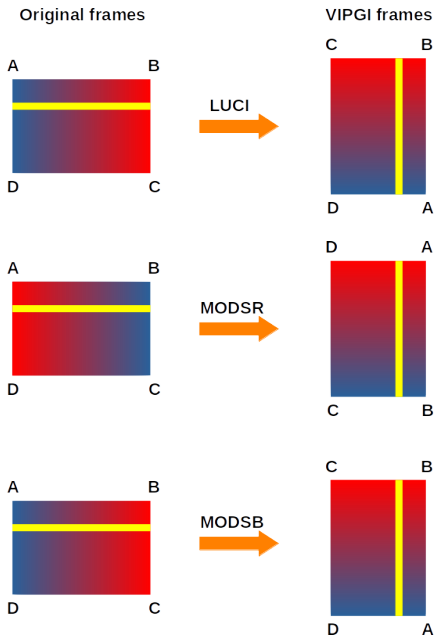

Fig. 5.2: This figure shows the new orientation of spectra after the ingestion in SIPGI. The solid yellow line indicates the target spectrum while the bluish/reddish pattern indicates the versus of wavelength scale.

- **Corrects LUCI files for non-linearity** Raw LUCI data are linearized. This means that ADU of raw frames (ADU_{raw}) are corrected to $ADU_{lin} = ADU_{raw} + k\,(ADU_{raw})^2$ with $k = 2.898 \cdot 10^{-6}$ for LUCI1 and $k = 2.767 \cdot 10^{-6}$ for LUCI2 (LBT staff, private communication).
- **Reads all of the keywords necessary to categorize the data**
- **Renames the files** The MODS LBT raw frames have the following format:

  mods*ArmFilter.yyyymmgg.????*.fits

  where *Arm* can be 1 or 2 depending on whether the data was taken on LBT arm 1 or 2, *Filter* can be r o b depending on the filter used, *yyyymmgg* is the observing data, and *????* is a four digits incremental number.

  Similarly, LUCI raw frames are in the form:

  luci*Arm.yyyymmgg.????*.fits.

  These names do not give much of an idea about the nature of the data contained within the file: whether the file is a bias, a flat, a scientific imaging or a spectroscopic observation.

  Once SIPGI acquires the file information from the keywords header, it re-names each file in the form:

  CC_target_grism_filter_m.fits

where CC is the type of data (*sc* for science, *ff* for flat field, *lp* for arc lamp images, *bs* for bias, *dk* for dark, *off* for lamp off), *target* is the target name, followed by a string that indicates the grism/filter. Finally, the file name ends with *m*, the exposure sequence number of the frame. In case of MOS observation, the mask ID is inserted too.

Example:

sc_MyQuasar_ID140777_G400L_Dual_001.fits

is a scientific frame of the target MyQuasar observed with the mask ID140777 using the grism G400L in Dual configuration. These new names simplify the browsing of the data.

- **Append calibration tables to files** As part of the organization step, a number of auxiliary files are appended to the raw FITS files. The built-in data organizer appends as FITS file extension the: *Grism table* (*GRS*), *CCD table* (*CCD*), *Lines catalog* (*LIN*) and *Mask file* (*MMS*). The *Paf file* content is stored in the Primary header of the file.

Table 5.1: The auxiliary files appended to each imported frame for LUCI observations.

LUCI frame	*CCD Table*	*Paf File*	*Grism Table*	*Lines Cat*
dark	✓			
slitless flat	✓			
throughslit flat	✓	✓	✓	
lamp	✓	✓		✓
science	✓	✓	✓	✓

Table 5.2: The auxiliary files appended to each imported frame for MODS observations.

MODS frame	*Mask File*	*CCD Table*	*Paf File*	*Grism Table*	*Lines Cat*
bias		✓			
slitless flat		✓			
throughslit flat	✓	✓	✓	✓	
lamp	✓	✓	✓		✓
science	✓	✓	✓	✓	

- **Copies the raw files in the Project directory and organize them in Datasets and Reduction Units** The rotated, renamed raw files, with all of the auxiliary files appended, are organized in the Project directory according to their type, PI, target name, and observing mode.

Caution: The user must **not** delete by hand data from the Project directory. If the user needs to remove a file, it must be done within SIPGI.

CHAPTER

SIX

THE SIPGI GRAPHICAL INTERFACE

The graphical interface is designed to quickly and easily browse the data at the various stages of the data reduction and to execute and check the various reduction steps. It is entirely written in Python to simplify the handling of graphical elements and to take advantage of the object-oriented programming capabilities offered by this language.

Fig. 6.1 shows the layout of the graphical interface as it appears before ingesting raw data. It is organized in three main operational area, highlighted by the three colored boxes: the starting and setting area (red box in Fig. 6.1), the data manager area (blue box in Fig. 6.1), and the reduction and analysis area (green box in Fig. 6.1).

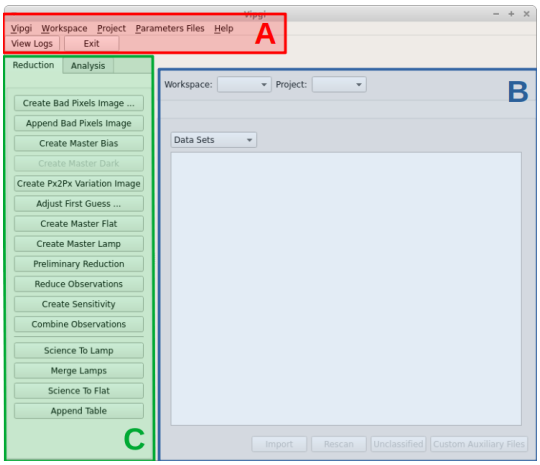

Fig. 6.1: The SIPGI layout at first run

6.1 The Starting and Setting Area

In the upper left region of the graphical interface (box A in Fig. 6.1) there is the Staring and Setting Area. It includes four menus, namely `SIPGI`, `Workspace`, `Project`, `Parameters` which control the main setting of the software plus the `Help` menu.

In the `SIPGI` menu the user has to set some general preferences as:

- the primary and secondary FITS files viewer. SIPGI offers the opportunity to visualize data with two different tools.
- the log level. There are six different log levels from *Off*, i.e. SIPGI does not provide any message during the execution of different reduction steps, to *Debug* the most detailed log level. A good compromise between information provided and verbosity is log level *Info*;

- Rescan the project directories at project loading. Activating this option, SIPGI rescans the Project directories at each start.

In the `Workspace` menu the user can:

- manage the creation of a new *Workspace* clicking on the option `New`;
- have information on the *Workspace* that is shown in the data manager area clicking the option `Info`;
- delete the *Workspace* that is shown in the data manager area clicking the option `Delete`.

The `Project` menu is similar to the *Workspace* one. Here the user:

- manage the creation of a new *Project* clicking on the option `New`.
- have info on the *Project* that is shown in the data manager area clicking the option `Info`;
- delete the *Project* that is shown in the data manager area clicking the option `Delete`. With the option `Delete`, the user can no longer work on his/her data within SIPGI, but the copy of the raw data created by the built-in data organizer (and possibly all the data products of reduction process) are still on the user disk in the Project directory;
- delete the *Project* that is shown in the data manager area **and** the files from the Project directory clicking the option `Delete All Data`.

> **Note:** The option `Delete All Data` removes both the *Project* **and** the copy of the data and products from the disk. **If no copy of the reduction products (including final ones) has been made, all will be lost!**

The `Parameters Files` menu allows the user to set the input parameters of any reduction recipes. The detailed behavior of each recipe is controlled via a set of parameters that are read in by the recipe at the beginning of its execution. SIPGI provides a default value for each parameter tuned on the reduction of LBT MODS and LUCI spectra. Default values change on the basis of the instrument.

Clicking on `Parameters Files` the list of the recipes appears. Clicking on one recipe, a panel with all its parameters is shown (see Fig. 6.2). Each entry has a brief description on the right side. Moving the mouse on the description, a more detailed description appears. Once all the changes have been done, the user can decide among different options with the buttons at the bottom:

- `Run`: it directly runs the recipe from the panel, but does not save the changes for next interactions;
- `Save`: it saves the changes but does not run the code;
- `Save and Run`: it saves the changes and runs the code;
- `Reset`: it sets the parameters values to their default values;
- `Cancel`: it closes the windows.

Any time the user saves the changes, they are valid throughout all of the *Workspace*.

The `Help` menu shows this manual within the browser.

The `View Logs` button opens the log panel.

The `Exit` button closes SIPGI.

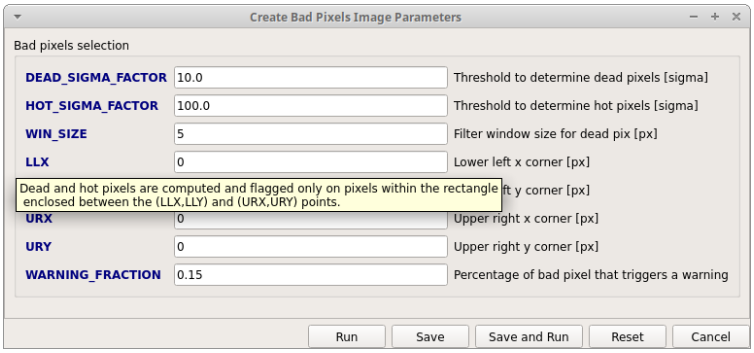

Fig. 6.2: The figures shows a `Parameters File` example. Each parameter is followed by a short description. Some parameters have a long description as shown by the yellow box. The long description can be visualized going with the mouse on the short description.

6.2 The Data Manager Area

In the data manager area (box B in Fig. 6.1) the user can control the data at different levels, from the ingestion of raw data in a new *Project*, to the selection of specific input to reduction recipes.

The data manager area is characterized by four buttons at its bottom, namely `Import`, `Rescan`, `Unclassified`, `Custom Auxiliary Files`, which control the ingestion of the data and the auxiliary files.

The `Import` button allows the user to put raw files under the SIPGI control. Once the user has created a new *Workspace* and a new *Project*, he/she has to import raw data in the SIPGI project, i.e. raw data must be processed by the built-in data organizer. At the end of this process, the user will find data organized in different *Datasets* in the central panel of the data manager. Concurrently, the user will find a copy of the raw files, organized in the SIPGI scheme, in the Project directory specified at the moment of the creation of a new *Project*.

The `Unclassified` button allows the user to see the list of raw files not imported by SIPGI. If some raw files do not have the appropriate entries in all of the keywords necessary to SIPGI (see *Appendix B: keywords necessary to SIPGI to import and categorize raw data*), the pipeline does not import them since it misses information to categorize and classify them. Similarly, SIPGI does not import those data for which it does not find the appropriate auxiliary files.

The `Rescan` button allows the user to import *new* files in SIPGI format in the current *Project*. This button is mostly for support purposes, i.e. it allows user to upload in SIPGI products produced by the Help Desk.

The `Custom Auxiliary Files` button opens the *Custom Auxiliary Files Manager*. During the reduction, the user can have the necessity to upload a *Customized auxiliary file* that better matches his/her scientific necessities as, for example, a *Spectro-photometric standard* at higher resolution or a new *Stellar template*. In addition the user could desire to reuse a *Master File* obtained in a old reduction for the reduction of a specific target. Once a new file is uploaded, it will be ingested in SIPGI that can use it for reduction purposes (for more details see the *Note*).

The central part of the data manager area allows the user to browse within his/her data. The two drop-down menus at the top of the area (see Fig. 6.3) permit to select the *Workspace* and the *Project*. Once selected, the *Datasets* associated to that *Project* appear in the central part of the data manager area.

Clicking on a specific *Dataset*, the user will have access to the different *Reduction Units* (see Fig. 6.4) and consequently to the data contained therein.

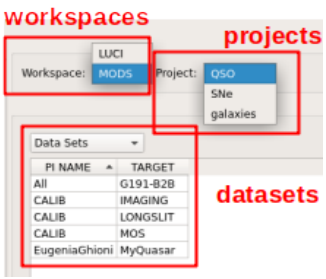

Fig. 6.3: The Figure shows the *Workspace* and *Project* drop-down menu and the aspect of the central part of the data manager area once raw files are imported.

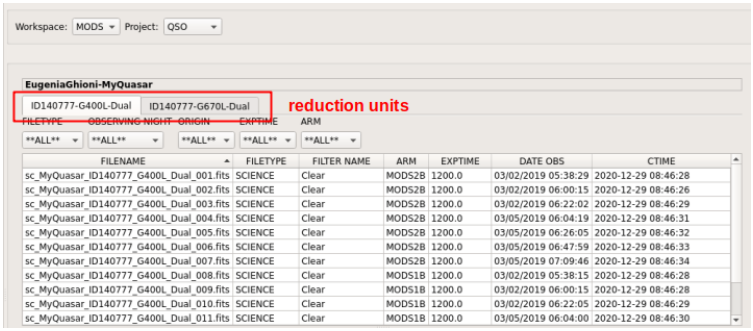

Fig. 6.4: The figure shows the organization of data produced by the built-in data organizer. In particular, it shows (red squares) two *Reduction Units* collecting the scientific data of the target MyQuasar observed with the mask ID140777 with the G400L-Dual grism and with the G670L-Dual grism. These data are in the *Project* QSO belonging to the *Workspace* MODS.

6.2.1 Reduction unit layout

Fig. 6.5 shows the typical aspect of a *Reduction Unit*. In each *Reduction Unit* there are three main panels: the frames panel, the master frames panel, and the generic master frames and auxiliary files panel.

6.2.1.1 Frames panel

In the top panel (**A** in the figure Fig. 6.5) the files of the *Reduction Unit* are shown. A series of drop down menus allows the user to filter the files in the *Reduction Unit* according to their file *type*, *observing night*, *origin*, *exposure time*, *dichroic*, *arm*, and in LUCI workspaces also according to *readmode*, *savemode* and *DIT* (these last three info are necessary for the creation of the *Master Dark*).

6.2.1.2 Master frames panel

In the middle panel (**B** in the figure Fig. 6.5) all the Master calibration files created in the current *Reduction Unit* such as Master Flat, Master Lamp, Master Dark, sensitivity function are shown. Every Master calibration file created by the user is visible only in the *Reduction Unit* where it is created. To make a Master File available to all the *Reduction Units* of the current *Projects*, the user must promote it to generic master frame (see `Share as auxiliary` in the *Context menus* section)

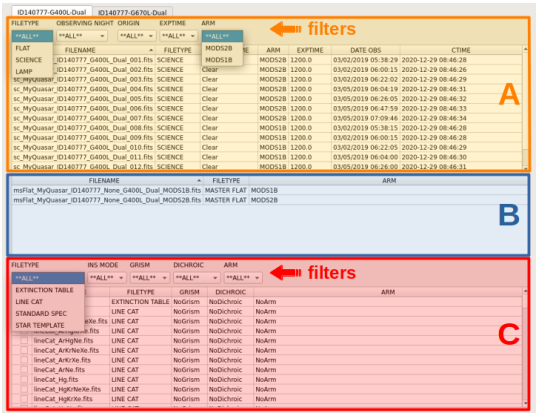

Fig. 6.5: This figure shows the *Reduction Unit* layout organized in 3 panels: **A** the frames panel, **B** the master frames panel, **C** the generic master frames and auxiliary files panel

6.2.1.3 Generic master frames and calibration files

In the bottom panel (**C** in the figure Fig. 6.5) of a *Reduction Unit*, SIPGI shows all the auxiliary system and customized files, plus all the Master Files that have been promoted by the user to the generic master frame panel. A set of drop-down menus on top of this panel allows the user to easily select files.

6.2.1.4 Context menus

The panels **A** and **B** have 2 context menus to easily handle the reduction unit files.

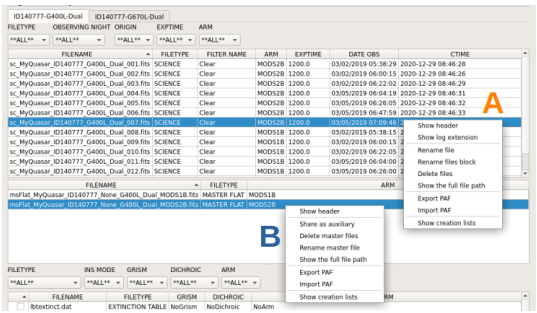

Fig. 6.6: The figure shows the context menus available in the panels **A** and **B** of the Reduction Unit.

For each file in the panel **A**, it is possible to select the following actions:

- `Show Header`: shows the header of the selected files;
- `Show log extension`: displays logs of the reduced frames (present if frames are reduced with the flag `DUMP_LOG=True`) (see *The data reduction recipes*);
- `Rename file`: renames the single selected file;
- `Rename files block`: renames a bunch of selected files, replacing the common part of the file name with something provided by the user;
- `Delete files`: deletes selected files;
- `Show the full file path`: shows the path on disk of the selected file;
- `Export/Import PAF`: this 2 entries allow to share the *Paf files* between files exporting and importing these values.

- `Show Creation list`: shows the list of files used to create the selected combined file or the selected bad pixel image.

Important: If the user wants to delete some files from the *Project* he/she **must** use the `Delete files` option and **do not delete** them from the Project directory.

The panel **B** provides the same context menu of panel **A** plus the entry `Share as auxiliary`. This entry allows to promote a Master File from the master frame panel to the generic one. This procedure will move the selected file from the middle **B** panel to the bottom **C** panel. In this way, the promoted file will be at disposal for every *Reduction Unit* of the *Project*.

6.3 The reduction and the analysis area

6.3.1 The Reduction Tab

The `Reduction` Tab contains two main blocks: in the top one there are the main reduction recipes and in the bottom one are present some utility functions. In each *Workspace*, only the recipes useful for the reduction of the corresponding instrument are activated. Those non active have light gray color buttons.

The buttons have a quite explanatory name and in the top block they are ordered following the main steps of a standard reduction.

The standard execution of each recipe is quite simple thanks to the file organization. The user must select the input files in the *Reduction Unit*, optionally the *Calibration files* and/or the *Master File* and click the recipe button. Some recipes require input files from different *Reduction Units*. For this reason their functioning passes through the creation of a input files list. The user can recognize these recipes from the presence of "..." in the button name.

The task performed by each recipe, its inputs and outputs are described in *The data reduction recipes*.

6.3.2 The Analysis Tab

The `Analysis` Tab contains a set of utilities to visualize and inspect the results. The input files and functioning are described in *The Analysis Utilities*.

CHAPTER

SEVEN

THE DATA REDUCTION RECIPES

Most of the recipes of the DRS are written entirely in C to ensure a satisfactory execution speed. Some recipes group together different reduction steps. This minimizes the number of recipes that one needs to execute to complete the data reduction process, while keeping the reduction blocks logically and operationally separated. This allows the astronomer to carry out intermediate checks on the quality of the data reduction process using the provided utilities (see *The SIPGI reduction cookbook*).

Each recipe is a standalone program, whose detailed behavior is regulated via a set of input parameters. SIPGI is released with a default configuration for all the input parameters which depends on the instrument (see *parameters files* and *Workspace*).

Important: We urge users to check these parameters and to set the configuration that best suits their reduction purposes.

The recipes outputs are FITS files, and to avoid increasing the number of produced files, the different reduction mid-products together with the various calibration tables useful for the reduction are appended as extensions to the original FITS files, instead of creating independent ones. This is not done to save disk space, but to simplify the astronomer's work, as the data reduction process would otherwise lead to the creation of thousands of files.

All of the pipeline recipes can be executed via a very simple point and click mode in the graphic interface, taking advantage of the files organization made by the built-in data-organizer. For practical examples see the *Cookbook section*.

7.1 Create Bad Pixels Image

It creates the bad pixel map which stores the information on the position of dead and hot pixels. This recipe is able to automatically detects 2 kinds of bad pixels: the dead and the hot pixels.

The recipe detects dead pixels using slitless flats. It combines the input frames using a median $F = median\{f_1, ..., f_n\}$ and creates a smoothed version of this image $\bar{F}$ using a median filter with a window size defined by the parameter `WIN_SIZE`. It divides the two frames $\hat{F} = F/\bar{F}$ and estimates the median value $m_{\hat{F}}$ and the standard deviation $s_{\hat{F}}$ of the image $\hat{F}$. In case of MODS observations, this operation is performed for each quadrant.

The parameter `DEAD_SIGMA_FACTOR` defines the detection threshold factor d: every pixel below $m_F - d \cdot s_F$ is labeled as dead.

If dark frames are provided (in LUCI reduction), the recipe also looks for hot pixels. Using a median, it combines dark frames and estimates the median value m_D and standard deviation s_D of this combined

image. The parameter HOT_SIGMA_FACTOR defines the threshold factor h: every pixel above the level $m_D + h \cdot s_D$ is labeled as hot.

All the pixels labeled as hot or dead are stored in the bad pixel image, i.e. an image with the good pixels values set at 1 and the bad pixels values set at 0.

The recipe offers the possibility to exclude the edges of the frame from the detection of bad pixels. In particular, median values, standard deviations and the detection of bad pixels is performed within the rectangle identified by the two points (LLX, LLY) and (URX, URY).

Table 7.1: **The Create Bad Pixels Image Recipe Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Slitless flat frames**	**Slitless flat frames and/or dark frames**
OUTPUTS	Bad pixel image	Bad pixel image

7.2 Append Bad Pixels Image

It loads bad pixels from the already created bad pixel image and overwrites the bad pixels list in the *CCD Table* of each frame. The user must append the bad pixel image to the frames to clean. If the UPDATE parameter is True, the recipe merges the list of bad pixels already present in the *CCD table* with the new one provided.

Table 7.2: **The Append Bad Pixels Image Recipe Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Frames to polish**;	**Frames to polish**;
	Bad pixel image	**Bad pixel image**
OUTPUTS	Input frames with updated CCD extension	Input frames with updated CCD extension

7.3 Create Master Bias

It creates the Master Bias, i.e. a FITS file containing the bias level to subtract to scientific frames. This recipe is available only for MODS. For LUCI reduction, the bias level is removed with the Master Dark (see *Create Master Dark* section for more details).

The recipe takes in input a list of bias frames. If desired, activating the flag CLEAN_COSMIC in the parameter file, the recipe polishes input biases from cosmic rays.

The cosmic rays polishing algorithm is similar to the one used by the MIDAS command FILTER/COSMIC.

The input image is smoothed using a median filter. The Poisson error P of this image is estimated. Then a loop is done on original pixels searching for cosmic rays candidates; a candidate is a pixel whose original value differs from the median filtered one by more than COSMIC_THRESHOLD $\cdot P$.

Contiguous candidates are then grouped together; each group is then processed to verify that it is not a real object. The recipe identifies the pixel of the group with the maximum count and the 8 ones around it. The mean M of the values of these 8 pixels is computed. A group is marked as a cosmic-ray hit if

COSMIC_RATIO $\cdot M$ is lower than the maximum value itself. The recipe returns an error if the percentage of cosmic rays is greater than COSMIC_MAX_FRACTION. Finally cosmic-rays hits are cleaned using the same algorithm as for the bad pixels cleaning (see *Create Master Dark*).

If the parameter CLEAN_NEGATIVE_PIX is True negative pixels are treated as cosmic rays.

After the cosmic ray cleaning procedure, frames are combined according to the selected COMB_METHOD:

- MEDIAN: median of input frames;
- AVERAGE: average of input frames;
- KSIGMA: the sigma clipped average of input frames. K_SIGMA_LOW and K_SIGMA_HIGH are the lower and upper sigma factors of the clipping;
- REJECT: for each pixel (x,y) of the input frames, computes a mean value after removing the lowest and highest values based on the percentages defined by the MIN_REJECTION and MAX_REJECTION parameters.

Finally, if the entry MOCK_BIAS is set to True, the recipe creates a mock Master Bias frame. This is an image full of 0 that doesn't remove the bias level from the frames, but is used by the *Preliminary Reduction* to trim the prescan/overscan regions.

The user can save the log messages in a separated extension of the Master Bias and/or can record all the files used as input flagging the DUMP_LOG and/or the DATA_LINAGE entry, respectively.

Table 7.3: **The Create Master Bias Recipe Summary**. In bold mandatory inputs

	MODS
INPUTS	**Bias frames**
OUTPUTS	Master Bias

Note: Together with Master Bias, SIPGI can estimate the bias level from the prescan/overscan regions of each frame. In this case, the bias level is not stored in a Master file but it is estimated on the fly for each frame during the *Preliminary Reduction*). In this case, SIPGI estimates eight bias levels: two for each quadrant, one for each quadrant channel. The single bias level is computed as the 1-sigma clipped average of the pixel values in the current channel/quadrant.

7.4 Create Master Dark

It creates the Master Dark, i.e. a FITS file with the dark level to subtract to scientific frames. This recipe is available only for LUCI. It is standard practice at LBT to provide a set of dark frames with the same DIT, NDIT, READMODE and SAVEMODE as the scientific frames (both science and telluric ones). The availability of dark frames with the same DIT as the scientific ones simplifies the reduction process, since they can be used to simultaneously remove both bias and dark levels. For this reason, LUCI calibrations do not include bias frames and, consequently SIPGI is not design to treat bias in LUCI *Workspaces*.

Important: Take in consideration that for a correct removal of bias and dark level using SIPGI for LUCI reductions, Master Dark must be obtained on frames with the same DIT and READMODE of scientific frames.

The recipe takes in input a list of dark frames d_i. If desired, activating the flag `CLEAN_COSMIC` in the parameter file, the recipe polishes input dark frames from cosmic rays as described in *Create Master Bias* section. If the parameter `CLEAN_NEGATIVE_PIX` is True, negative pixels will be handled as cosmic rays by the recipe and cleaned at the same time. After the optional cosmic ray correction, frames are combined using the `COMB_METHOD` provided by the user (see *Create Master Bias* section for more details on the `COMB_METHOD`).

If the `NDIT_NORM` flag is activated, the recipe normalizes input dark frames by their NDIT value (retrieved from the header) $\bar{d_i} = d_i / NDIT_{d_i}$ before combining them.

Note: Remember that $t_{exp} = DIT \cdot NDIT$

When this normalized Master Dark is used in the reduction process (see e.g. *Preliminary Reduction*), the recipe automatically multiplies it by science NDIT values, before subtracting it. This allows the user to subtract the dark level in case of science frames with the same DIT, READMODE, SAVEMODE values but different NDIT with respect to the dark frames.

If the entry `CLEAN_BAD_PX` is True, the recipe polishes the combined image from bad pixels contained in the CCD table. The bad pixel correction replaces the bad pixel value interpolating the values of the first good pixels around the bad pixel. For recipe details see the *Appendix E*

Table 7.4: **The Create Master Dark Recipe Summary**. In bold mandatory inputs

	LUCI
INPUTS	**Dark frames**
OUTPUTS	Master Dark

7.5 Create Pix2pix Variation Image

This recipe uses flat slit frames obtained without any mask (e.g. slitless flat frames) to create an image which stores the information on the pixel to pixel variation. In the MODS case this image is also used to correct the different gain responses between the 4 MODS quadrants.

The recipe takes as input slitless flats. In MODS reduction, if requested, it subtracts the bias level from the input frames using the `BIAS_METHOD` provided by the user. The choice is between: `SKIP`, `MASTER`, and `PRESCAN`. With the `SKIP` option the user skips the bias subtraction. In case of `MASTER` choice, the user must also supply the Master Bias. This is the canonical subtraction of the Master Bias for each frame. With the `PRESCAN` method the bias level is computed from the prescan/overscan regions of the same input flats as described in *Create Master Bias*. In LUCI reduction, if a *Master Dark* is provided, the recipe subtracts the dark level from flat frames.

If the `CLEAN_COSMIC` entry is set to True, the recipe polishes frames from cosmic rays. After this, the recipe combines flat frames with the `COMB_METHOD` provided by the user. For more details on the cosmic cleaning and the combining methods see *Create Master Bias*.

A smoothed version of the combined image is created quadrant by quadrant using the `SMOOTH_METHOD`. The recipe has two smoothing method: a `MEDIAN` filter and a `GAUSSIAN` kernel. The median filter replaces the value of each pixel with the median estimated in a box of size `SMOOTH_SIZE_BOX` centered on that pixel. The `GAUSSIAN` method smooths the image using a Gaussian kernel with the sigma defined by `SMOOTH_SIGMA` (more details about this algorithm can be found in scipy.ndimage.gaussian_filter).

MODS shows different gain levels for each quadrant as shown in the Fig. 7.1.

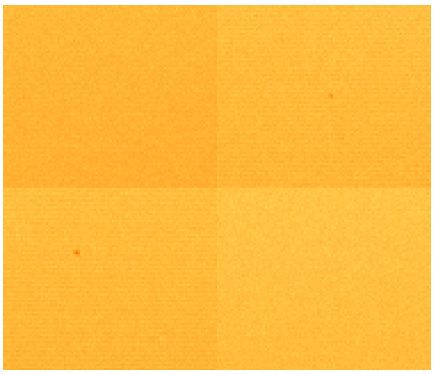

Fig. 7.1: A zoom on a flat frame at the center of the MODS detector.

These differences can be corrected using the parameter `GAIN_WIN_SIZE`. If `GAIN_WIN_SIZE` is greater than zero, the recipe selects for each quadrant of the smoothed image a square region at the quadrant junction (see Fig. 7.2) and collapses it along the cross dispersion direction. An example of the collapsed signal for the 4 quadrants is shown in Fig. 7.3 (see dashed lines).

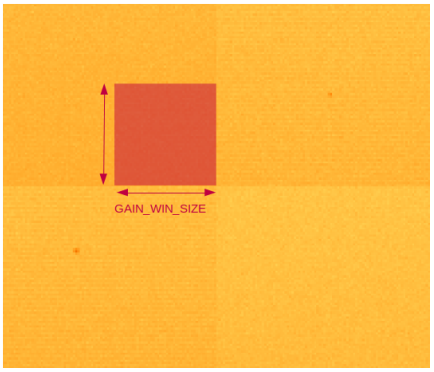

Fig. 7.2: The red square is the `GAIN_WIN_SIZE` region used to compute the normalization factor.

Then it fits a second order polynomial to model the flat variation (solid lines in the Fig. 7.3). The values of these polynomial in the junction are the scaling factor applied to each quadrant (dots in the Fig. 7.3).

This image stores the large scale variation of the pixels response. Finally, the original combined image is divided by the smoothed gain-corrected image in order to recover the pixel to pixel variation map.

This frame is an optional input for the *Create Master Flat* recipe. If it is not provided, pixel-to-pixel variation will be directly computed on the through-slit flats provided as input to the recipe. Although it is not mandatory, it is relevant in two cases:

- In MODS reduction, both in LS and MOS observations, the slit(s) pattern on through-slit flats does not allow a proper correction for different responses of different quadrants (keep in mind that even the MODS LS observations present 5 slits). The Pix2Pix Variation Image provides this correction.
- In LUCI reduction, it offers the possibility to correct for the pixel-to-pixel variation in all of the cases in which through-slit flats are not aligned with science frames.

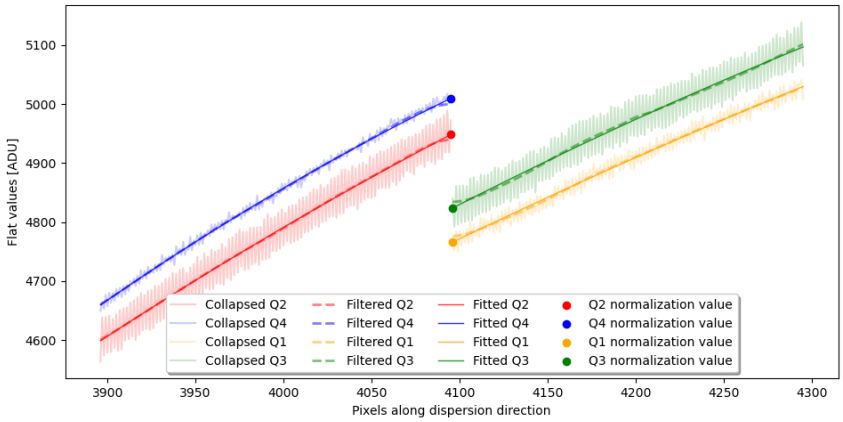

Fig. 7.3: An example of the procedure for the 4 different gain levels adjustment in MODS data. The jagged lines are the collapsed flat signal along cross dispersion direction in the original frame; each line highlights the different channels response which affects the MODS data. The dashed lines are the collapsed flat signal in the smoothed images; the solid thin lines are the second order polynomials fitted to the dashed lines; the dots are the value of this polynomial at the junction edges. These values are used as normalization levels.

Table 7.5: **The Create Pix2Pix Variation Image Recipe Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Slitless flat frames**	**Slitless flat frames**
OUTPUTS	Pix2Pix variation image	Pix2Pix variation image

7.6 Adjust First Guesses

This is an interactive task, which allows the user to refine on real data the theoretical *OPT*, *CRV* and *IDS* Models stored in the *Paf file*. For more details see *adjust first guesses* paragraph.

The task requires in input a flat field frame and a lamp frame, in its standard execution. For more details on peculiar cases see *the Cookbook*.

The task loads the *Lines catalog* appended by the organizer to the lamp frame and selects lines in the extraction range defined by the *WLEN START* end *WLEN END* keywords of the *Grism table* (contained into the flat frame).

Using the Models solutions contained in the header of the flat field frame, the task creates DS9 regions for the expected line positions and spectra edges and displays them on the selected lamp frame.

The user can check the current Models solutions and, in case they do not fit the emission lines pattern on the lamp frame or the spectra edges, he/she can "adjust" the solutions moving the DS9 regions on the real line position/edge and refitting new solutions. The goal of this interactive process is to obtain the final Models solutions that best matches the real lines/edges from a visual point of view. These final solutions will be considered as "first guess" for the more sophisticated recipes *Create Master Flat* and *Create Master Lamp* which compute the definitive spectra location and the wavelength solution, respectively.

Attention: The task uses lamp frame to check and adjust the solution contained into the flat frame. This solution is then applied to science frames to extract 2D spectra and wavelength calibrate them. For this reason, the position and the tracing of dispersed spectra in all of the three kinds of frames (flat, lamp, science) must match (see *adjust first guesses* for more details in case this condition is not verified).

Table 7.6: **The Adjust First Guess Recipe Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Through-slit flat or science frames**	**Through-slit flat or science frames**
	Through-slit lamp	**Through-slit lamp**
OUTPUTS	Input flat or science with updated Models	Input flat or science with updated Models

7.7 Create Master Flat

This recipe creates the *Master Flat*, a FITS file that contains:

- the information on the slits location and on the spectral tracing on the raw frames;
- the pixel to pixel variations image.

The recipe takes in input a (list of) through-slit flat(s) (or science frames, see *cookbook*). In MODS reduction, if requested, it subtracts from frames the bias level according to the `BIAS_METHOD` provided by the user (see *Create Pix2pix Variation Image* for more details on the methods). In LUCI reduction, if a Master Dark is provided, dark level subtraction is performed. After this, if the `CLEAN_COSMIC` entry is set to True, the recipe polishes frames from cosmic rays according to the `COSMIC_RATIO` and `COSMIC_THRESHOLD` settings, otherwise it directly combines the frames with the `COMB_METHOD` provided (see *Create Master Bias* for more details on cosmic rays detection/polishing and on the combining method). The combined image is corrected for bad pixels (see *Create Master Dark* section).

The accurate tracing of the spectra curvatures is performed fitting the edges of the spectra on the combined and polished frame. Starting from the position of the edge defined by the first guesses, the recipe computes the real location of the spectrum left edge. The main steps of the edges computing are:

1. SIPGI uses the solution of CRV Model obtained by the adjust procedure as first guess for the left edge location;
2. in order to increase the S/N of the spectrum edge, the recipe stacks together multiple rows. In this case, it moves along this edge and cuts thumbnails of the spectrum centered on the first guess value and with dimensions `TRACING_BIN` ·`EXTRA_PIXELS` (blue boxes in Fig. 7.4);
3. the software collapses these thumbnails along the dispersion direction and computes the edge profile (the red curve in the figure) in each thumbnail.
4. using the profile, SIPGI measures the edge position as the peak of the numerical derivative of the profile;
5. these edge values are used by SIPGI to fit a 1D polynomial which describes the spectra curvature along the dispersion direction.

Once fitted the left edge of each spectrum in the frame, the recipe determines their right edges by shifting the left ones by the slit length (quantity read from the header of the file). The right edge position can be

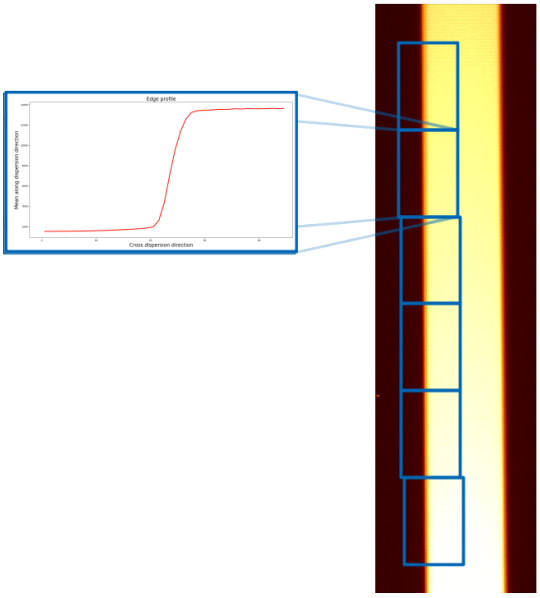

Fig. 7.4: This figure shows how the master flat recipe works on the edge of the stacked flat frame. Blue boxes indicate the flat region collapsed by the recipe to maximize the edge signal that is shown in the inset box (red curve). The dimension of these boxes is set by the two parameters `TRACING_BIN·EXTRA_PIXELS`. The center of the box is defined by the "first guesses" stored in the *Paf file* of the input flat.

refined by activating the entry `RIGHT_EDGE_REF`. In this case, the recipe uses the determined right edges as first guesses and proceeds as for the computation of the left edges. If the distance between the two edges so estimated is less than the slit length, the solution is saved, otherwise the recipe takes as solution the first guess. The tracing solution will be stored in a new extension of the Master Flat, the Extraction Table (*EXR*). This extension contains the grism table header taken from input files, the best-fitted solution of the tracing and the first guesses of the IDS Model (the last ones will be refined with the *Create Master Lamp* recipe).

The master flat tracing limits (in pixels) are defined by the keywords: *LLEN LO* and *LLEN HI* stored in the *Grism Table* header. In standard condition tracing limits include extraction range limits (see *grism table note*). However distortions can be such that this condition is no met (the *Show Spectra Location* utility allows the user to check this). This implies that the wavelength solution will be computed only within the tracing limits. To avoid this, user can push the tracing limits of `ADJUST_TRACING_LIMITS` pixels over the extraction limits.

After the spectra tracing, the recipe computes the pixel-to-pixel variation. If a Pix2Pix Variation Image is provided, the recipe substitutes the image of the combined flat with it. In this case the user can skip all the setting parameters in the box "Pixel to pixel variation". If the Pix2Pix Variation Image is not provided, the recipe will compute the pixel-to-pixel variation on the combined through-slit flat following the same approach described in *Create Pix2pix Variation Image*. Starting from the tracing solution stored in the EXR extension, the recipe extracts the spectra and smooths their image using the method defined by `SMOOTH_METHOD`. This is a `MEDIAN` or `AVERAGE` smoothing box filter, with box size defined by the `SMOOTH_BOX_SIZE` parameter. Then it obtains the pixel 2 pixel map slit by slit, dividing the original spectra by their smoothed version.

Important: If the user decides to derive the pixel-to-pixel variation from through-slit flats, some caveats should be remembered:

- for MODS data, this approach does not remove the different responses of the four quadrants (for

more details see *Create Pix2pix Variation Image*);

- flat field frames and science must be aligned.

If through-slit flats and science frames are not aligned (a quite frequent case in LUCI observations), the user can decide to perform the spectral tracing directly on science frames. In this case the pixel-to-pixel variation cannot be computed on input frames and the user has two possibilities: either he/she provides in input a Pix2Pix Variation Image or performs a mock Master Flat clicking the entry `MOCK_FF`. In the last option, the Master Flat image has all the values set to 1, thus the file stores the information on the tracing but not on the pixel-to-pixel variation.

Both the Pix2Pix Variation Image and the Create Master Flat recipe address only the issue of the pixel-to-pixel variation. None of them takes into account the large scales variations on the frames. Large scales variations along the dispersion direction will be compensated by the sensitivity function. Our recipes do not take into account for different responses of the detector along the spatial direction. **For this reason, it is strongly recommended to plan observations with targets and telluric/standard in the same position in order to minimize the effects.**

Table 7.7: **The Create Master Flat Recipe Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Through-slit flat or science frames**	**Through-slit flat or science frames**
	Master Bias	Master Dark
	Pix2Pix variation image	Pix2Pix variation image
OUTPUTS	Master Flat	Master Flat

7.8 Create Master Lamp

The recipe starts from the information contained in the *Master Flat*, namely the spectra position and tracing, and refines on real data the wavelength solution predicted by the first guesses and stored in the EXR extension of the Master Flat. SIPGI allows the user to compute the wavelength solution both on arc lamp frames and science frames (see *Adjust First Guesses*). Science frames can be very useful in case instrument distortions on calibration frames do not match science frames ones.

The Master Lamp is fundamental to achieve the 2D spectra extraction: the solution contained into this frame allows to obtain 2D spectra corrected by the distortions and linearly calibrated in wavelength.

The recipe starts by cleaning up the input frames from the instrument signature. In MODS reduction, if requested, it removes the bias level accordingly with the `BIAS METHOD` (see *Create Pix2pix Variation Image* for more details). In LUCI reductions, if a Master Dark is provided as input, the recipe subtracts dark current from the input frames. Then it polishes the input frames for effects like cosmic rays, if the parameters `CLEAN_COSMIC` and `CLEAN_NEGATIVE_PIX` are activated (see *Create Master Dark*).

Starting from these polished frames, the recipe uses the solution contained into the EXR extension of the Master Flat to locate the 2D spectrum on the CCD. If the `EXTRA_PIXELS` parameter is greater than 0 the recipe refines the spectrum location within a searching window of `EXTRA_PIXELS` size. For each 2D spectra (i.e. for each slit), the CreateMasterLamp recipe cuts the 2D spectrum in N slices: one slice per pixel along the cross dispersion direction. Namely, if the 2D spectrum is 11 pixels large, the recipe will create 11 slices of the 2D spectrum (see Fig. 7.5).

The recipe moves along each slice following the spectral curvature defined by the Master Flat and it handles each slice like a 1D spectrum. Moving along the slice it computes the **real** line positions. Then

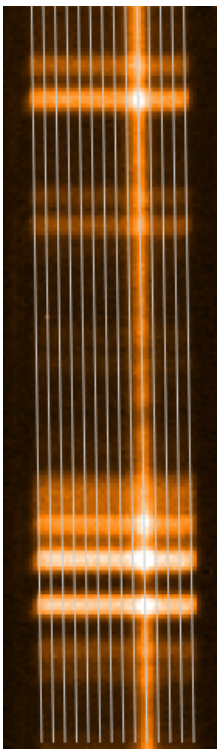

Fig. 7.5: This picture shows the slicing of the 2D spectrum performed to compute the wavelength solution on the 2D spectrum. The wavelength solution is computed in each 1px slice of the 2D spectrum.

it computes the **slice wavelength solution** using these measured lines positions.

More in details, for each slice the following iterative process is carried out:

1. the recipe picks the nominal lines positions from the input *Lines catalog*. It uses every line in the catalog in the wavelength range defined by the *WLEN START* end *WLEN END* keywords in the Extraction Table;
2. it uses the *IDS Model* (in the first iteration the one stored in the Master Flat) to have a first guess in pixel of the position of the given lines.
3. it measures the *real* line position by computing the barycenter of the flux detected in a window of dimension `EXTR_WINDOW` centered on the expected line position (see Fig. 7.6). If the line measurement fails (e.g line too flat, ...) the line is flagged as *lost*, otherwise it is considered *good*;
4. it computes a new *IDS Model* using only *good* lines; all the lines deviating more than 2.5 sigma from the best-solution are flagged ad *bad*;
5. if no *bad* lines are found, the recipe exits the iterative loop;
6. if *bad* lines are found, the recipe repeats the loop using the new *IDS Model* as "first guess". This procedure is repeated till the recipe finds the same *bad* lines in two consecutive loops;
7. in case the number of *good* lines is lower than twice the IDS polynomial order, the slice is flagged as *invalid*.

Typically, different arc lamps (Xe, Ne, Ar) are acquired. In case the emission lines of a single lamp does not cover the whole spectral range of science frames,multiple lamp frames can be merged together. This could be done before running the Create Master Lamp recipe using the utility *Merge Lamps*. In this case a dedicated *Lines catalog* must be provided to the recipe.

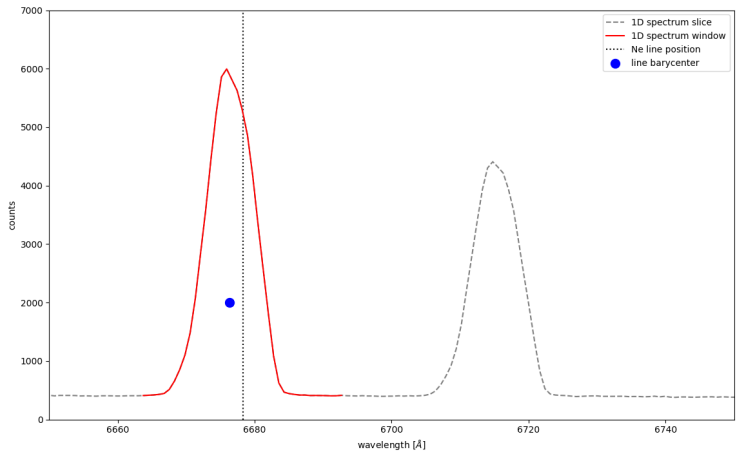

Fig. 7.6: Zoom in of a single slice spectrum (grey dashed line) around the spectral region where a line is expected (vertical dotted line; from the *Line Catalog*). The position of the observed line (full blue circle) is derived as the barycenter of the spectrum within a window of center the first guess position of the lines and width defined by the EXTR_WINDOW parameter (red line). This barycenter position is compared to the expected position to derive the correction needed for the wavelength calibration.

The final solution is stored in the Extraction Table extension.

Table 7.8: **The Create Master Lamp Recipe Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Through-slit lamp or science frames**	**Through-slit lamp or science frames**
	Master Flat	**Master Flat**
	Master Bias	Master Dark
OUTPUTS	Master Lamp	Master Lamp

7.9 Preliminary Reduction

This recipe takes in input a Master Flat and raw frames (optionally a Master Bias/Dark). It corrects input frames for bad pixels and cosmic rays, removes dark/bias level, and applies the flat field correction.

For MODS reduction, if requested, the recipe subtracts the bias level according to the `BIAS_METHOD` (see *Create Pix2pix Variation Image* for more details). In LUCI reduction, if a Master Dark is provided, the recipe subtracts the dark level. After this, it trims the prescan/overscan regions from input frames. If the keyword `CLEAN_COSMIC` is activated, it polishes raw frames from cosmic rays (see *Create Master Bias* for more details) while `CLEAN_BAD_PIX` activates the polishing of bad pixels.

The pixel-to-pixel variation will be corrected according to the image stored in the Master Flat primary header.

Note: If a mock Master Flat is provided, no pixel-to-pixel variation will be performed.

The output frames will have the same name of the raw frames with a final extension indicating the type of correction applied:

- D for dark subtraction;
- B for bias subtraction;
- F for flat field correction (actually since Master Flat is mandatory, the "F" suffix will be always present, also in case of a mock Master Flat);
- C for cosmic rays treatment.

As example, the file raw_file_name_BFC.fits is the raw_file_name.fits file bias subtracted, flat fielded and with cosmic rays treated.

The BFC file contains all the extensions of the input file, but the *Primary* extension now contain the polished frame. If ERROR_MAP is True, the BFC file will contain also an ERROR extension including the error on the polished frame. For more details on the errors propagation see *Appendix C*.

Table 7.9: **The Preliminary Reduction Recipe Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Raw science frames**	**Raw science frames**
	Master Flat	Master Flat
	Master Bias	Master Dark
OUTPUTS	Pre-reduced frames	Pre-reduced frames

7.10 Reduce Observations

This recipe extracts 2D spectra wavelength calibrated, free of distortions, optionally sky-subtracted and flux-calibrated, determines their extraction profiles and extracts the 1D spectra.

The recipe takes in input the pre-reduced frames and a Master Lamp.

If CHECK_DITHER is activated, the first step performed by the recipe is to control whether the input frames are dithered one respect to the other before to move on. This is an important cross-check to be performed when ABBA or DAVIES method are used (see *later*).

7.10.1 Refining the spectral tracing & wavelength solution on science frames

The pipeline gives the possibility to refine the slits position and the wavelength calibration using the sky lines in the science frame. If the USE_LINES parameter is activated, the recipe checks on each science frame the spectral tracing and the wavelength solution stored in the Master Lamp. In details:

- the recipe reads from the EXT table of the Master Lamp a series of sky lines that it will use for the two checks;
- using the solution stored in the Master Lamp, it determines the edge position of the 2D spectrum at the position of these sky lines;
- it recomputes the edge location starting from these positions and comparing them to the real ones in a window with width EXTRA_PIXELS;
- it computes the shifts between the old and new edge positions (for all sky lines, in all slits), derives the median value of these shifts, and applies a shift along the cross dispersion direction (X axis) equal to the median value to the spectral tracing solution stored in the Master Lamp.

A similar procedure is followed to refine the wavelength calibration. The recipe computes the shift between the sky lines position predicted by the Master Lamp and the one re-computed on real data. The fitting procedure looks for the line in a window of `LINE_WIDTH` dimension, slice per slice, on each slit (like SIPGI does to create the *Master Lamp*). Then the recipe estimates the median value of all of these shifts and applies a shift along the dispersion direction (Y axis) equal to median value to the EXR solution.

Both adjustments are done twice, in the first iteration the recipe computes the shifts and in the second one it checks the new solution and applies a further (smaller) refinement.

Since the wavelength solutions are computed slice by slice the solution along the slit can be slightly jagged. If `FIT_IDS_COEFFS` is positive the recipe applies a smoothing polynomial fit (of order `FIT_IDS_COEFFS`) to the model coefficients, in order to reduce this effect. In case `FIT_IDS_COEFFS` is negative no smoothing is performed. This parameter could be also used to repair *invalid* slices (see *Create Master Lamp*). The IDS model of such a slice is extrapolated from the smoothed fit.

7.10.2 Extraction of 2D spectra

The extraction of the 2D spectra is a set of procedures that start from the pre-reduced frames (in which the dispersion runs on the y-axis), re-bins them and provide in output the 2D spectra rectified by distortion and wavelength calibrated (hereafter EXR2D spectra) (see Fig. 7.7 and for more details *Appendix D*). In the EXR2D spectra extraction, SIPGI also "flips" the spectra: the dispersion direction is aligned along the x-axis with wavelength increasing from left to right, and cross dispersion is aligned along y-axis. This product is stored in a new extension called *EXR2D*.

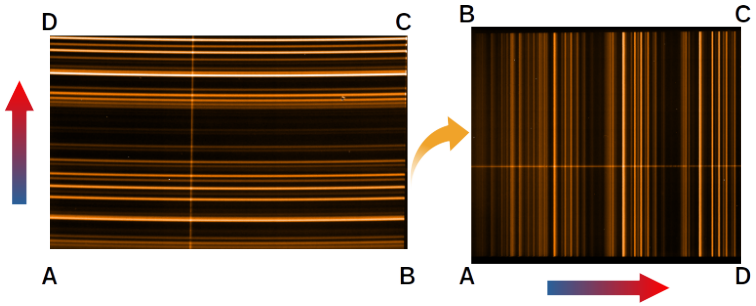

Fig. 7.7: Scheme showing the transformation carried out during the extraction of the 2D spectra. In this transformations, the spectrum is flipped, wavelength calibrated and corrected by distortions.

The recipe extracts the EXR2D spectra in the range [*WLEN START*, *WLEN END*] with a *WLEN INC* sampling. By default, these values are read from the Master Lamp. The user can change the extraction limits and the spectral sampling at run-time, using the parameters: `WLEN_START`, `WLEN_END`, and `WLEN_INC`.

SIPGI is able to extract EXR2D spectra before or after the sky subtraction, depending on the user's requests (see *next paragraph*).

7.10.3 Sky subtraction

SIPGI can perform sky subtraction directly on raw frames and on EXR2D. Sky subtraction on raw frames is suggested only on spectra considered distortions-free, i.e. for raw frames in which lines of constant wavelength are perfectly orthogonal to the dispersion direction. The pipeline distinguishes between *"distorted"* and *"un-distorted"* spectra on the basis of the parameter `SLIT_TOLERANCE`. SIPGI defines *"un-distorted"* those spectra for which the maximum drift in pixels of the IDS solution along the cross dispersion direction is lower than the `SLIT_TOLERANCE` parameter. If the `SLIT_TOLERANCE` is negative, SIPGI will consider all of the slits as *"distorted"*.

SIPGI offers two main procedures for the sky subtraction. These procedures can work both on raw frames (for the *"un-distorted"* spectra) and on re-binned EXR2D (for the *"distorted"* spectra).

- The `SKY_METHOD`: in this case the sky level is computed (and subtracted) row by row on the pre-reduced frames or column by column in EXR2D spectra using the method defined by the parameter `SKY_METHOD`. The choice is between:
  - `SKIP` the recipe does not subtract the sky level;
  - `MEDIAN` the recipe computes the median of input pixels values;
  - `FIT` the recipe performs a fit of `POLY_ORDER` order to the pixels values;
  - `WFIT` the recipe performs a weighted fit of `POLY_ORDER` order to pixels values.

  In all the methods, the recipe excludes from the sky-level estimate the pixels interested by the objects signal. These pixels are identified using the objects identification procedure described *below* applied on not sky-subtracted frames.

  The user can decide to also exclude from the sky-level estimate the `SLIT_MARGIN` pixels near the spectrum edges. If `RSLIT_MARGIN` is positive, the recipe excludes `SLIT_MARGIN` pixels from the bottom (left in EXR2D) edge and `RSLIT_MARGIN` pixels from the top (right in EXR2D) edge. In the sky level estimate, whatever the method, the recipe excludes from the computation all of the pixels occupied by (detected) sources. On the *"un-distorted"* slits, the recipe allows to control the minimum number of pixels to be used for sky subtraction through the parameter `N_SKY_MIN`.
- The `ABBA` method: it is activated flagging the parameter `ABBA`. In this case the recipe removes the sky level subtracting from each frame the next dithered frame in the exposure sequence. The exposures must be dithered in order to avoid the source removal. Even in this case the *A-B* subtraction is performed on raw frames for *"un-distorted"* slits and on EXR2D spectra for *distorted* slits

Hint: If the PI wanted to apply ABBA on raw frames for all slit, he/she can set a huge insane value of `SLIT_TOLERANCE`, i.e. 1000.

In the LUCI *Workspaces* the `DAVIES` method is provided too[1]: it can be activated flagging the parameter `DAVIES`. If `DAVIES` is selected, all slits are treated as *"distorted"* by SIPGI. As the `ABBA` method, `DAVIES` is based on the subtraction of two dithered exposures (see Davies 2007 for more details).

If the `DAVIES` method is active two additional parameters can be used to control the sky subtraction: `DAVIES_THERMAL_SUB` and `DAVIES_SCALE_FACTORS`. If `DAVIES_THERMAL_SUB` is True *SIPGI* fits a black-body function to the thermal background in the sky spectrum, and subtracts it (see Davies 2007). If `DAVIES_SCALE_FACTORS` is True, the recipe computes a scaling factor for each vibrational transition band of the OH spectrum.

[1] The Davies's algorithm is an implementation of an IDL script kindly provided by Giovanni Cresci.

Caution: ABBA and DAVIES methods cannot be used at the same time.

SKY_METHOD and ABBA or DAVIES can be mixed according to the PI needs. In Fig. 7.8 and Fig. 7.9 two schemes of all the sky subtraction possibilities offered by SIPGI are reported.

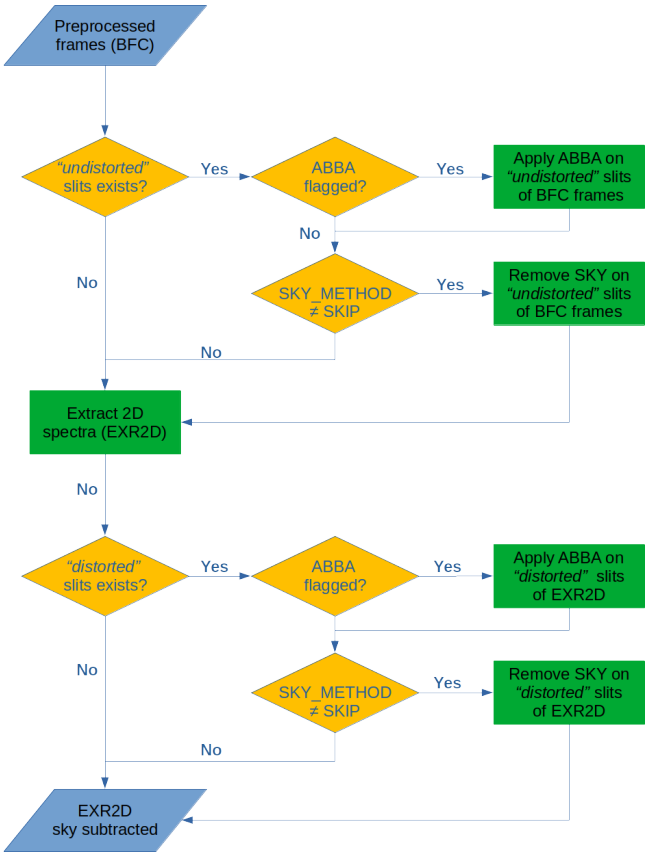

Fig. 7.8: This diagram shows the sky subtraction flow in case the ABBA method is selected.

Caution: If DAVIES parameter is true all slits will be considered *"distorted"*.

In MODS reduction, if an *Extinction Table* is selected, the extinction correction is performed during the 2D extraction.

7.10.4 Fringing Correction

If FRINGING_CORR is True, the recipe corrects input frames for the fringing pattern.

Caution: SIPGI uses a very simple approach for fringing correction. Moreover, the procedure is applied on wavelength calibrated frames (EXR2D). Users are strongly encouraged to evaluate if the correction procedure is suitable for their purposes.

To correct for the fringing pattern a global residual image is computed from the sky-subtracted EXR2D frames. In each frame the areas covered by the (detected) objects are masked; these areas can be enlarged

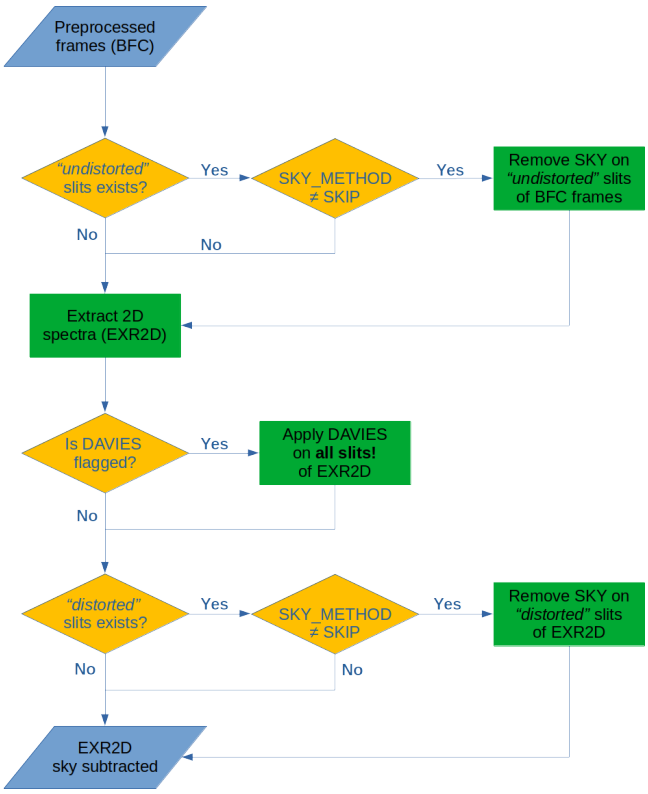

Fig. 7.9: This diagram shows the sky subtraction flow in case the `DAVIES` method is selected.

using the `FRING_PIXELS` parameter which determines the number of pixels that are added at each side.

If the `FRING_INTERP` parameter is False, the pixel in the objects areas will be ignored in the following combination step. If `FRING_INTERP` is True, a linear interpolation is computed in each column between the values at the edges of the areas; the pixel values of the objects area are then substituted with the interpolated values (that are used in the combination step).

The resulting objects-masked images are combined using a median filter to estimate the global residual image; this image is then subtracted from all the sky-subtracted images to remove the fringing pattern.

Important: It is not recommended to use the fringing correction in case of objects that overlap in all exposures (i.e. no or small offsets)

7.10.5 Flux Calibration & 1D extraction

Once the sky subtraction and the wavelength calibration have been carried out, the recipe moves to the flux-calibration. SIPGI offers the possibility to have:

- 1D spectra and EXR2D spectra not calibrated in flux (if no sensitivity function is provided);
- flux calibrated 1D spectra and EXR2D spectra not calibrated in flux, if the sensitivity function is provided and the parameter `CALIB_EXR2D` is False;
- 1D and EXR2D spectra calibrated in flux if a sensitivity function is provided and parameter `CALIB_EXR2D` is True.

Table 7.10: Scheme of the Reduce Observations recipe output units on the basis of input provided. Column 1: Is the sensitivity function provided?; Column 2: setting of the CALIB_EXR2D parameter; Column 3: unit of the 1D spectra; Column 4: unit of 2D spectra.

Sens. Funct.	CALIB_EXR2D	Output 1D unit	Output EXR2D unit
No	False	ADU	ADU
Yes	False	erg cm^{-2} s^{-1-1}	ADU
Yes	True	erg cm^{-2} s^{-1-1}	erg cm^{-2} s^{-1-1}

The extraction of 1D spectra requires the estimate of the extraction profile. This procedure relies on the EXR2D spectra (sky-subtracted and flux calibrated following the user's choices), and computes the profile following these steps:

- for each row i of the EXR2D the recipe sorts the values of all pixels;
- in case SPEC_WSTART and SPEC_WEND parameters are provided, the recipe sorts pixels between SPEC_WSTART and SPEC_WEND. This allows to select a specific spectral region for sources detection (see Fig. 7.10).
- given this sorted array, the recipe computes the average (m_i) on the central fraction of the array defined by SPEC_FRACT. This allow to reject outliers.
- m_i as a function of spectrum row is the extraction profile (see Fig. 7.11);
- the *mean* and the *std* of the profile are estimated. All profile points above *mean* + *std* · DETECTION_LEVEL are defined as possible object candidates.

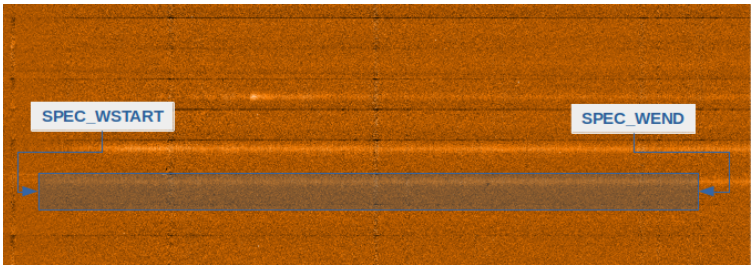

Fig. 7.10: The Figure show the behavior of the SPEC_WSTART and SPEC_WEND parameters on an EXR2D frame. If these parameters are activated, the recipe collapses the 2D spectrum regions highlighted in the blue box.

If the number of contiguous points is greater than MAX_OBJ_SIZE, the recipe tries to deblend the detection in more sources. If the number of contiguous points is between MIN_OBJ_SIZE and MAX_OBJ_SIZE, the recipe considers the detection as a single object; if the number of contiguous points is lower then MIN_OBJ_SIZE, it rejects the detection. If no detection is found, and REFINE_DETECTION is True, the recipe tries to identify sources with a parabolic fit of the extraction profile. All the information on the detected object are stored in the *Window Table*.

Once the detections have been identified, the recipe extracts the 1D spectrum. The recipe sums up the signal in all of the rows identified through the analysis of the extraction profile. It is to highlight that the 1D extraction procedure in SIPGI does not take into account any curvature in the object trace. For this reason, for a good extraction, it is crucial that the object signal is parallel to pixel rows(see Sec. 4.1).

Note: The user can activate an Horne extraction (Horne 1986) flagging the parameter HORNE_EXTRACTION **only if the objects trace in 2D spectra are perfectly aligned with the pixels rows**.

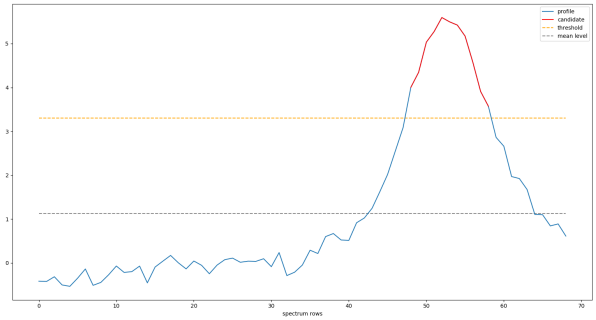

Fig. 7.11: The Figure shows the extraction profile (blue line). The grey dashed line indicates the mean value of the profile, the yellow dashed line the threshold, and the red line the spectrum rows where a spectrum is expected to be (red line does not starts from yellow line due to the finite pixel dimension). In case the detection is confirmed (see `MIN_OBJ_SIZE` and `MAX_OBJ_SIZE` parameter), the recipe collapses all of the rows highlighted in red.

If the object trace is not perfectly aligned with the pixel rows, the Horne extraction is compromised.

If `ERROR_MAP` is True, the recipes will propagate the errors for all the steps of the reduction (i.e. sky subtraction, extraction of 2D and 1D spectra) . For more details on the errors computation see *Appendix C*. If the `ERROR_MAP` flag was also activated during the *preliminary reduction*, the recipe also takes into account the contribution to the error derived from the preliminary reduction.

The output frames **replace** the pre-reduced input frames and their names are modified by adding an *R* as final extension. As example, the file `raw_file_name_BFC.fits` is **replaced** by the `raw_file_name_BFCR.fits` file which contains - among other things - the preliminary and fully reduced 1D and 2D spectra.

According to the choices made by the user different extensions will be appended to the output file:

- *Primary*: the preliminary reduced frame.
- *ABBASUB*: the sky values subtracted directly on the preliminary reduced frames by the ABBA method.
- *ABBASUB_ERROR*: the error estimate on the *ABBASUB* extension.
- *ABBASUBRAW*: the preliminary reduced frame ABBA subtracted
- *ABBASUBRAW_ERROR*: the error estimate on the *ABBASUBRAW* extension.
- *SKYSUB*: the sky values subtracted directly on the preliminary reduced frames.
- *SKYSUB_ERROR*: the error estimate on the *SKYSUB* extension.
- *SKYSUBRAW*: the preliminary reduced frame sky subtracted
- *SKYSUBRAW_ERROR*: the error estimate on the *SKYSUBRAW* extension.
- *WIN*: the window table. It contains geometry information about slits and about detected object (see the *Show Window Table* paragraph for more details)
- *EXR2D*: 2D extracted spectra wavelength calibrated and corrected by distortions.
- *EXR2D_ERROR*: the error estimate on the *EXR2D* extension.

- *SKY2D*: the subtracted sky on EXR2D.
- *SKY2D_ERROR*: the error estimate on the *SKY2D* extension
- *ABBA2D*: the subtracted sky on by the ABBA method.
- *ABBA2D_ERROR*: the error estimate on the *ABBA2D* extension
- *DAVIES2D*: the sky subtracted by the Davies procedure.
- *EXR2D_FRINCOR*: The EXR2D extension fringing corrected
- *EXR2D_FRINCOR_ERROR*: the error estimate on the *EXR2D_FRINCOR* extension.
- *EXR2DFLUX*: 2D extracted spectra flux and wavelength calibrated, corrected by distortions.
- *EXR2DFLUX_ERROR*: the error estimate on the *EXR2DFLUX* extension.
- *EXR1D*: 1D extracted spectra wavelength calibrated
- *EXR1D_ERROR*: the error estimate on the *EXR1D* extension.
- *EXR1DFLUX*: 1D extracted spectra wavelength and flux calibrated
- *EXR1DFLUX_ERROR*: the error estimate on the *EXR1DFLUX* extension.
- *SPH2D*: the Sensitivity-function applied on data (see *next paragraph*)

The EXR2D extension, in each of its declinations (e.g. EXR2DFLUX, EXR2D_FRINCOR), contains the 2D extracted spectra of all the slits except alignment and reference slits (see Fig. 7.12). The 2D extracted spectra of "scientific" slits are shown all together in the EXR2D extension and re-arranged from bottom to top of the image according to their increasing SIPGI slit number (for an example, see Fig. 7.10).

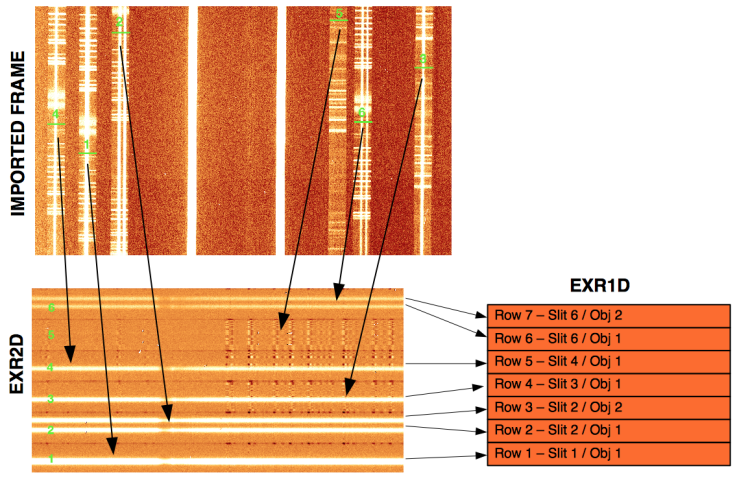

Fig. 7.12: An example of a raw imported frame (top panel) for a MODS MOS case, and of the EXR2D (bottom left panel) and EXR1D (bottom right panel) extension in its corresponding reduced frame. Green lines and numbers in the raw imported frames indicate the slits positions and their relative SIPGI slit numbers, respectively (see the *Show Slit Position* utility for more details). SIPGI does not enumerate reference and alignment slits.

In MOS data, SIPGI enumerates slits following the slit numbering order in the LMS/MMS file (see Fig. 7.12, top panel). SIPGI does not enumerate alignment and reference slits (by default square slits are considered reference slits).

SIPGI enumeration starts from #1. The relation between the SIPGI and the LMS/MMS slit numbering is described by the set of keywords *LBT INS SLIT# ID* in the header of the files. For example: *LBT INS SLIT1 ID = '07_M31'* means: the first slit extracted by SIPGI (*SLIT1*) is the slit number *07* in the LMS/MMS file and its original object name is *M31*.

Note: Since the SIPGI slits numbers follow the slit numbering order of the LMS/MMS file, the order of the 2D spectra sequence in the EXR2D extension from the bottom to the top of the frame could not match the sequential order of 2D spectra from left to right in the imported frame (see Fig. 7.12). For example, if two slits located at the mask extremes have two sequential numbers in the LMS/MMS file, their 2D spectra will be close each other in the EXR2D, even if other "scientific" slits are present between them on the mask. To know the SIPGI slit number of each slit in imported frames, and hence identify the corresponding 2D spectra in the EXR2D extension, the user can use the *Show Slit Position* utility. For more details see also the *Show Window Table* section.

The EXR1D (or the EXR1DFLUX) extension is an image in which each row is a 1D spectrum. From the bottom to top, the image contains the 1D spectra of all the detected objects in the same order these objects are located in the EXR2D image, starting from the slit number 1 (i.e. the one at the bottom of the frame) (see Fig. 7.12).

Table 7.11: **The Reduce Observations Recipe Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Pre-reduced frames**	**Pre-reduced frames**
	Master Lamp	**Master Lamp**
	Sensitivity Function	Sensitivity Function
	Extinction Curve	
	"Sky" frames (for ABBA method)	"Sky" frames (for ABBA/DAVIES methods)
OUTPUTS	Reduced frames	Reduced frames

7.11 Create Sensitivity

This recipe creates the *sensitivity function*, i.e. the function which converts spectra from ADU to erg cm^{-2} s^{-1-1}. It takes in input the wavelength calibrated standard/telluric spectra (see *Reduce Single Observations*), compares them with the relative *Stellar template*/*Spectro-photometric standard* and computes a curve as function of lambda which perform the conversion.

Given the different observing strategy for MODS and LUCI observations, the recipe follows slightly different approaches for the two instruments.

7.11.1 MODS sensitivity function

The recipe takes in input the wavelength calibrated spectra of the standard star, the relative *Spectro-photmetric standard* and optionally the relative *Star template*.

Starting from the wavelength calibrated frames of the standard star (e.g. BFCR files sky-subtracted but non flux calibrated) the recipe takes the 1D extracted spectra of the star. Since reduced frames can have many detections, the recipe identifies the star detection as that with the highest counts and takes the relative 1D spectrum. However spurious detections could be mis-classified by the recipe, for this reason, it is good practice that the user identifies the star detection and provide it to the recipe. Once 1D spectra of standard star have been identified, the recipe stacks these spectra together using a median.

Then it reads the input *Spectro-photometric standard*. If the parameter MAG is True, the recipe assumes that the *Spectro-photometric standard* is expressed in magnitudes, otherwise it assumes the *Spectro-photometric standard* in fluxes units. The *Spectro-photometric standards* released with SIPGI cover the full MODS wavelength range. However, the user can provide a customized *Spectro-photometric standard* with a shorter wavelength coverage with respect to the data. Considering this, the recipe offers to the user the opportunity to provide in input a *Stellar template* (of the same spectral class of the observed standard star) that is used to extend the "shorter" customized *Spectro-photometric standard*. If a template is provided, the recipe re-bins the template to the same binning of the *Spectro-photometric standard*, scales it to the same flux level of the *Spectro-photometric standard* at the HEAD_JOIN (or TAIL_JOIN) point and substitutes the *Spectro-photometric standard* with the template at wavelength blueward (or redward) the HEAD_JOIN (or TAIL_JOIN). If HEAD_JOIN or TAIL_JOIN parameters are negative, the recipe will not join the template and the *Spectro-photometric standard*.

After this, the recipe interpolates the *Spectro-photometric standard* to match the sampling of the observed stars and divides the median spectrum of observed star for the *Spectro-photometric standard* (possibly extended) creating a *raw sensitivity function*. If EDIT_ABS is True, the recipe edits the raw sensitivity function replacing its values in the four wavelength ranges [6820-6960] , [7150-7350] , [7580-7750] , and [9280-9850] with straight lines that join the extremes of the ranges. Finally the recipe smooths the raw sensitivity. If the parameter METHOD is SMOOTH, the recipe smooths the raw sensitivity function using a median filter of size WIN_SIZE and then if the parameter MEAN_WIN_SIZE is greater than zero, it refines the smooth with a mean filter of size MEAN_WIN_SIZE. If the METHOD is SPLINE the recipe fits the raw sensitivity function with a B-spline. The kind of spline is defined by the parameter SPLINE_MODE, the function computes a B-splines of order k=SPLINE_MODE+1. If SPLINE_MODE is 0, a cubic spline (k=4) is computed (for more details see GSL documentation). The user can provide the break points of the spline with the parameter SPLINE_BREAK. Otherwise the recipe uses the parameter SPLINE_ORDER to uniformly break the spline (as described by the gsl_bspline_alloc help).

7.11.2 LUCI sensitivity function

Differently from MODS observations, the recipe that derives the sensitivity function for LUCI data is more complex and foresees a different approach for LS and MOS data. The first steps done by the recipe are aimed at identifying the spectral type and the magnitude of the observed star.

This can be done in automatic by the recipe, if TARGET_NAME=NULL. In this case the recipe retrieves the telluric name looking for the *OBJECT* keyword in the observed frames (e.g. HIP-56147). If FILTER=AUTO and STELLAR_TYPE=NULL, the recipe uses this name to extract from the *Hipparcos Catalog* the spectral type and the nominal magnitude of the star. In the Hipparcos Catalog provided by SIPGI there are many magnitude for a given telluric. The recipe automatically retrieves the nominal magnitude in the filter closer to the one of the observations (f_{obs}). Once magnitude and spectral type have been identified, the recipe scales the the *Stellar template* corresponding to the telluric spectral type to the nominal magnitude of the star in the filter f_{obs}.

Note: This approach requires that the *OBJECT* keyword of the input files is filled in with the Hipparcos name of the telluric (i.e. HIP-56147 not HD-99966), and that the observed telluric is present in the Hipparcos Catalog. If the *OBJECT* keyword is not properly filled in the user **must** provide it to the recipe as `TARGET_NAME` = HIP 56147 (or just 56147).

If the telluric is not present in the Hipparcos Catalog, the user **must** provide:

- the `STELLAR_TYPE`
- the `MAG_VALUE`
- the `FILTER` in which the `MAG_VALUE` is estimated.

These operations produces the "normalized" template that is interpolated in order to match the sampling of observed data and then is compared with the observed star.

MOS LUCI sensitivity function

The standard practice for telluric observations in LUCI MOS program is to observe the telluric through the blue-most and red-most slit of the mask to maximize the spectral coverage. This means that to obtain a Sensitivity function which covers the whole spectra range, the recipe joins the spectra obtained in the two slits. For this reason in the MODS case the parameter `RED_AND_BLUE` must be True.

The joining procedure is driven by the parameter `AUTO_RED_AND_BLUE`. If it is True, the recipe tries to automatically recognize the blue-most and red-most 1D input spectra. If the entry is False, the user **must** provide input frames in even number *n* and the first *n/2* frames of the input list are assumed to be the blue-most spectra and the other *n/2* frames are assumed to be the red-most spectra. Once blue-most and the red-most spectra have been recognized, the recipe computes the two median spectra and join them at the point defined by the parameters `LAMBDA_JOIN` or `PIXEL_JOIN`:

- if `LAMBDA_JOIN` is positive it defines the joining point in ;
- if `LAMBDA_JOIN` is negative, but `PIXEL_JOIN` is positive the joining point is defined in pixels;
- if both the parameters are negative the recipe joins the two spectra at the middle point.

If the parameter `RED_AS_MASTER` is True, the recipe normalizes the blue-most spectrum to the red-most at the joining point and join the spectra, otherwise the recipe will normalize the red-most spectrum to the blue-most.

Once obtained the joined median observed spectrum, the recipe divides it by the "normalized" template obtaining a raw sensitivity function.

In the LUCI case the smoothing of the sensitivity function is driven by the parameter `METHOD`. If the `METHOD` parameter is `FIT` a polynomial FIT of order defined by `POLY_ORDER` if fitted producing the sensitivity function. If the `METHOD` is `INTERP` the input raw sensitivity function is divided in a number of intervals of length `INTERP_LEN` []. For each interval the median value of the raw sensitivity function is computed, then a Cubic spline is interpolated between median values of these intervals. If the parameters `LAMBDA_MIN` and `LAMBDA_MAX` are zero, the spline is interpolated along all the points of the raw sensitivity function. Otherwise this range can be limited by the parameters `LAMBDA_MIN` and `LAMBDA_MAX`.

LS LUCI sensitivity function

For LS observations, the parameter `RED_AND_BLUE` must be False since just one slit is available. In this case, the recipe computes the median of all 1D spectra detected as the star. Then it divides the median spectrum by the "normalized" template obtaining a raw sensitivity function and smooth it like in the *LUCI-MOS case*

Table 7.12: **The Create Sensitivity Recipe Summary**. In bold mandatory inputs

	MODS	LUCI
INPUTS	**Wavelength calibrated frames**	**Wavelength calibrated frames**
	Spettro-photometric standars	**Stellar template**
	Stellar template	
OUTPUTS	Sensitivity function	Sensitivity function

7.12 Combine Observations

This recipe combines 2D spectra contained in the `BFCR` / `DFRC` frames and yields:

- stacked 2D spectra for each slit
- 1D calibrated spectrum for each detected object
- error estimated spectra for each product.

It takes in input at least two `BFCR` / `DFRC` frames and optionally a sensitivity function. The recipe combines the *EXR2D* extensions of input frames or *EXR2DFLUX* in case they were calibrated. If `USE_EXR2D_FRINCOR` is True the recipe combines the *EXR2D_FRINCOR* extensions. Combine methods are defined by the parameter `COMB_METHOD`. Available options are: `AVERAGE`, `MEDIAN` and `KSIGMA` (for details on methods see *Create Master Bias*). EXR2D spectra can be normalized by their exposure time using the entry `EXPTIME_NORM`.

Before combining frames the recipe shifts them if offsets are provided.

7.12.1 Compute Offsets

The recipe offers different methods to compute offsets. They can be selected in the `Combine Observation` panel, which appears at the recipe launch. The choices are:

- `Get from Headers`: the recipe computes offsets using RA and Dec information contained into the files header;
- `Compute from WIN`: the recipe uses the *OBJ_POS* of the detected object (in the window table) to estimate relative shift between frames. If some slits contain multiple detection, the recipe evaluates the distance between these detections and checks that it does not change between frames within an error defined by `MULTI_DET_STD`. The user can set a minimum number of common detections by the parameter `MIN_OBJ_OFFSET`, as well as, the slits to be used for the computation (`SLITS`);
- `Compute from EXR2D`: the recipe recomputes the objects detection on the *EXR2D* extensions as in the *Reduce Observations*. Then it performs the same computation of `Compute from WIN`.

The user can also compute the offset manually and provide them in the dedicated panel (see *Combine Observations* paragraph in the Cookbook section).

7.12.2 1D extraction

The recipe recomputes object detection on 2D stacked frames as the *Reduce Observations* recipe does. If the parameter `ALLOW_NO_DETECTION` is True and no detection is found the recipe exits without errors.

In case the Sensitivity function is selected, and the input frames are not already flux calibrated the recipe calibrates the 1D extracted spectra.

Table 7.13: Scheme of the Combine Observations recipe output units on the basis of input provided. Column 1: Is the sensitivity function provided?; Column 2: unit of input EXR2D spectra; Column 3: unit of the 1D combined spectrum; Column 4: unit of 2D combined spectrum.

Sens. Funct.	Input EXR2D unit	Output 1D unit	Output EXR2D unit
No	ADU	ADU	ADU
No	erg cm^{-2} s^{-1-1}	erg cm^{-2} s^{-1-1}	erg cm^{-2} s^{-1-1}
Yes	ADU	erg cm^{-2} s^{-1-1}	ADU

Table 7.14: **The Combine Observations Recipe Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Reduced BFCR frames**	**Reduced DFCR frames**
	Offsets between exposures	Offsets between exposures
OUTPUTS	Combined frame	Combined frame

7.13 Other data reduction utilities

7.13.1 Science to Flat

This button allows the user to convert a science frame into a flat frame that can be read by the *Adjust First Guesses* recipe and by the *create Master Flat*. Sometimes LS/MOS flats are not perfectly aligned with science frames. This implies that if the user adopts a flat frame to refine the OPT and CRV Models, he/she obtains a Master Flat in which the Extraction table does not well describe the location of the spectra in science frames. In this situation, the user can decide to use the same science frame as flat.

Important: If the user uses such kind of flat to create the *Master Flat*, the Master Flat should be MOCK, unless a dedicated *Pix2Pix Variation Image* is provided.

7.13.2 Science to Lamp

This button allows the user to convert a science frame into a lamp frame that can be used as input lamp by the *Adjust First Guesses* and to *create a Master Lamp*. This facility is provided to compensate misalignment between science and lamps (*science to flat*).

Bright objects in the science frames can affect the solution of the Master Lamp: the fit of the sky line positions can be biased by the object. To avoid this problem, the user can select two science frames with the object in 2 different positions and SIPGI will create a lamp frame object free.

By default the task appends to the output lamp the *Lines catalog* found in the first input frame. A custom *Lines catalog* can be provided to this recipe.

Table 7.15: **The Science to Lamp Utility Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Through-slit science frames**	**Through-slit science frames**
	Line catalog	Line catalog
OUTPUTS	Lamp frame	Lamp frame

7.13.3 Merge Lamps

It allows to sum together different lamps and to append to the resulting file a *Lines catalog* provided by the user. This task is useful to create a lamp which covers the whole lambda range to use in the *adjust first guesses* step.

Table 7.16: **The Merge Lamps Utility Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Through-slit science frames**	**Through-slit science frames**
	Line catalog	**Line catalog**
OUTPUTS	New lamp frame	New lamp frame

7.13.4 Append Table

It allows user to append: *Lines catalogs*, *CCD Table* or *Grism Table* to files. In case table already exists it will be overwritten without any warning.

7.13.5 SIPGI To Molecfit

During his/her own reduction, the user could decide to correct his/her spectra for atmospheric absorption features using the software Molecfit (see ESO).

SIPGI offers three utilities to interact with Molecfit: SIPGI to Molecfit, *Apply Molecfit Tra* and *Revert Molecfit Tra*

The SIPGI to Molecfit utility "exports" reduced data out of the SIPGI environment so that the user can process them with Molecfit. In addition to this, this utility creates all the necessary files to run the standalone version of Molecfit 2.0.1. In particular:

- *the param file*. SIPGI fills in the configuration parameters file with all the information that can be acquired by the FITS header of the reduced file. The user **must** fills all the other entries on the basis of his/her own reduction purposes.
- *the "exclude_p" file*. This is an empty file that the user can customize;
- *the "exclude_w" file*. This is an empty file that the user can customize;
- *the "include" file*. This file contains the regions expressed in μm to be processed by Molecfit. By default, SIPGI fills this file with the full default extraction range.

We remand to Molecfit documentation for more details on the software functioning.

7.13.6 Apply Molecfit Tra

Once the user has estimated the transmission function with Molecfit, this utility allows user to apply it to his/her spectra (1D and 2D, if present) *in SIPGI*.

7.13.7 Revert Molecfit Tra

Remove the Molecfit transmission function previously applied and restore the frame to their original form.

CHAPTER

EIGHT

THE ANALYSIS UTILITIES

8.1 DS9

This button opens SAOImageDS9 and by default displays the Primary FITS extension of every FITS file selected. The user can display a custom extension using the dedicated pull-down menu. Several files can be displayed in DS9 as different ds9-frames.

8.2 Primary Fits Viewer

It displays the selected file. The external software used to show the FITS is the one selected in the *SIPGI Preferences* as primary FITS viewer.

8.3 Secondary Fits Viewer

It displays the selected file. The external software used to show the image is the one selected in the *SIPGI Preferences* as secondary FITS viewer.

Hint: We suggest to select as Primary Fits Viewer your favorite tool to manage images, and as secondary your favorite tool to manage tables.

8.4 Show Slit Position

It shows the reference slit positions on the selected frame. SIPGI displays in DS9 the selected frame with green line regions showing the position where the slits are supposed to be according to the *Paf file* contained in the selected frame. For MOS observations, it shows the SIPGI number of each slit (see Fig. Fig. 8.1 and for more details about mask slit numbering see note below).

In case a reduced file is provided, the slit numbering is overlaid on the EXR2D extension,

Note: In MOS data, SIPGI enumerates slits following the slit numbering order in the LMS/MMS file. SIPGI does not enumerate alignment and reference slits (by default square slits are considered reference slits).

SIPGI enumeration starts from #1. The relation between the SIPGI and the LMS/MMS slit numbering is described by the set of keywords: *LBT INS SLIT# ID*. For example: *LBT INS SLIT1 ID = '07_M31'*

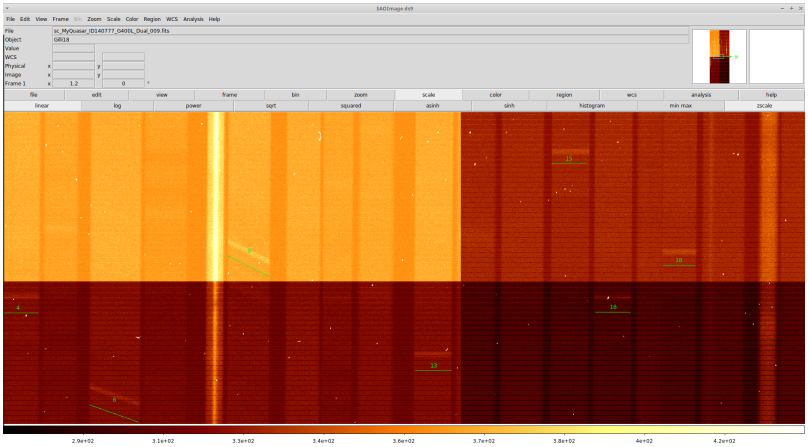

Fig. 8.1: A MOS frame as shown by the Show Slit Position utility. Green lines indicate the supposed slit position (in case of no distorted frame). The green numbers indicate the slit number.

means: the first slit extracted by SIPGI (*SLIT1*) is the slit number *07* in the LMS/MMS file and its original object name is *M31*.

EXR2D frames show 2D extracted spectra packed together from bottom to top according to their slit number .

Table 8.1: **The Show Slit Position Utility Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Through-slit flat frame**	**Through-slit flat frame**

8.5 Show Spectra Location

It shows the spectra location and spectral tracing using the solution stored in the *Master Flat*. The user can use this tool to check the quality of the tracing. A good tracing is necessary to properly follow the traces of the targets and extract them correctly. In fact, 1D spectra are extracted within a region that runs parallel to the pixel rows. This means that if the tracing presents some sort of curvature wrt the real edge of the 2D spectrum (i.e. wrt the pixel rows), the target will go out from the extraction region. The user must select the Master Flat to check and the frame on which this solution must be checked. SIPGI opens the frame in DS9 with superimposed the edges of the spectra (see white vertical "lines" in Fig. 8.2).

The extension of the lines identifies the spectral region fitted by the Master Flat and it is defined by the keywords PIX_LOW and PIX_HIGH contained in the *Grism tables*.

Black solid lines in the upper and lower part of the frame indicate the expected position of the extraction limits (i.e. WLEN_START and WLEN_END, see *Grism tables*) according to the solution stored in the Master Flat. These lines should be placed **within** the tracing region, since SIPGI will perform the wavelength calibration using all the lines included in the [WLEN_START, WLEN_END] range, but in any case never outside the tracing regions.

The utility `Show Spectra Location` can be used also on science frames. This allows the user to directly check the spectral tracing on science raw frames.

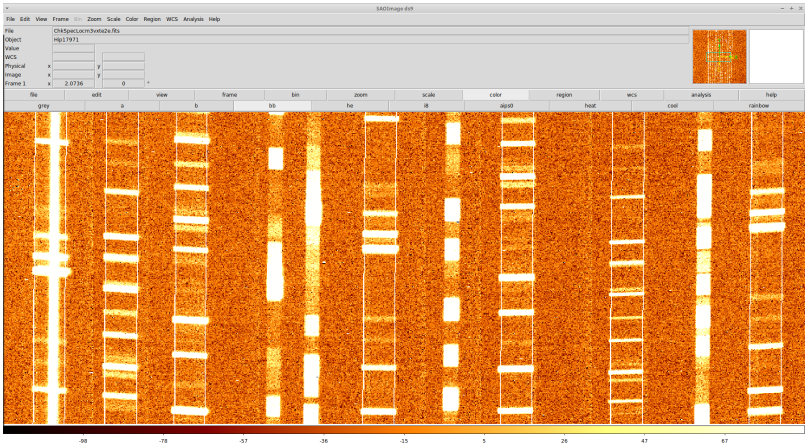

Fig. 8.2: A MOS frame as shown by the Show Spectra Location utility. White horizontal lines indicate the right and left edges of the slits as estimated by the Create Master Flat recipe.

Table 8.2: **The Show Slit Position Utility Summary**. In bold mandatory inputs

	MODS	LUCI
INPUTS	**Through-slit flat frame**	**Through-slit flat frame**
	Master Flat	**Master Flat**

8.6 Show Lambda Calibration

The *Master Lamp* frame contains the final solution used to obtain rectified 2D spectra linearly calibrated in lambda. For each slit, this solution predicts the wavelength to associate at each pixel in the frame.

This utility takes in input a Master Lamp (or a master Flat if the user wants to check the intermediate solution). In addition to the information shown by *Show Spectra Location*, it displays the position of the *Lines catalog* lines appended to the input frames, on the basis of the solution stored in the Master File (see Fig. Fig. 8.3). In case no additional frame is provided the recipe displays the solution directly on the master frame image.

The *Lines catalog* can be passed as further input. In case no *Lines catalog* is provided, the recipe will look for a *Lines catalog* in the input frames. If no catalog is found in the input frames, the utility uses the Master Lamp *Lines catalog*.

Table 8.3: **The Show Lambda Calibration Utility Summary**. In bold mandatory inputs

	MODS	LUCI
INPUTS	**Through-slit flat frame or science** (mandatory if Master Flat is provided)	**Through-slit flat frame or science** (mandatory if Master Flat is provided)
	Master Flat or Master Lamp	**Master Flat or Master Lamp**

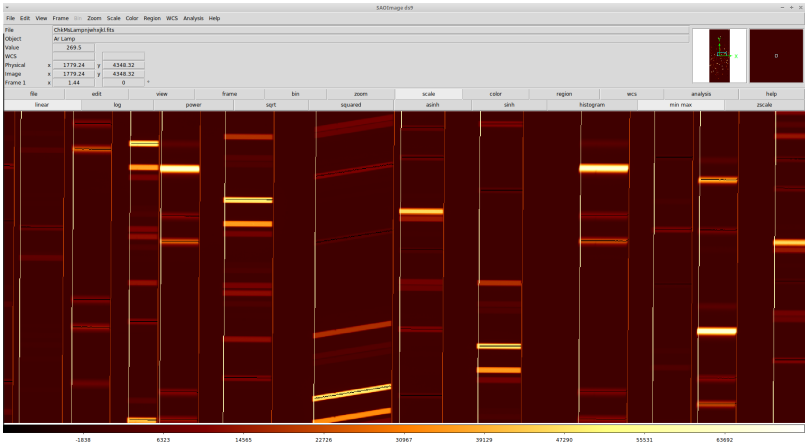

Fig. 8.3: A MOS frame as shown by the Show Lambda Calibration utility. Black lines show the lines expected positions as estimated by the Create Master Lamp(or Create Master Flat) recipe.

8.7 Check Lambda Calibration

This utility takes in input a Master Lamp and provides a quantitative estimate of the wavelength calibration accuracy.

For each slit it recomputes the real lines positions in the middle of the slit, and estimates the differences between the lines positions expected according to the input Master Lamp and real ones for all the lines in the catalog (*DeltaL* []). It shows a message like:

Slit 1 - good lines: 20; mean DeltaL=0.018640; DeltaL rms=0.129130,

containing:

- slit number;
- number of good lines (see *Create Master Lamp*) used for the fit;
- the mean and rms value of these differences.

Together with this, it provides a global information on the wavelength solution giving:

- the Median of the "mean DeltaL" over all the slits;
- the Median of the "DeltaL rms" over all the slits. A warning is raised when this quantity is greater than 1/5 of the *spectral sampling*;
- the Maximum DeltaL rms over all the slits;
- the Minimum DeltaL rms over all the slits.

For each line in the *Lines catalog*, the recipe provides also the number of slits in which the line is *good* (see *Create Master Lamp*), plus the median and std of the differences between the expected and real position of that line in all of the slits.

Table 8.4: **The Check Lambda Calibration Utility Summary**.
In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Master Lamp**	**Master Lamp**

8.8 Plot Lambda Calibration

This utility allows to check the wavelength calibration of the selected Master Lamp with deeper detail wrt the *Check Lambda Calibration*. The *Check Lambda Calibration* utility provides information on the quality of the wavelength calibration for each slit and on the global wavelength calibration quality. In both cases, all the information is picked up in the middle of the slit.

The user could be interested to have information on the wavelength calibration quality in the target position, and/or along the dispersion direction.

The `Plot Lambda Calibration` utility addresses both issues. It is a graphical task which allows:

- to have information on the wavelength calibration accuracy in any "slice" of the 2D spectra *for all the slits* (see *Create Master Lamp* recipe);
- to exclude some lines from the fit and refit the wavelength solution;
- to have information on the calibration accuracy for each fitted line.

For more details on its use, see the *Plot Lambda Calibration* description in the Cookbook section.

Table 8.5: **The Plot Lambda Calibration Utility Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Master Lamp**	**Master Lamp**

8.9 Plot Sensitivity

This utility allows the user to visualize the raw and smoothed selected sensitivity functions (see *Create Sensitivity*) and to check its quality.

For more details on its use, see the *Plot Sensitivity* description in the Cookbook section.

Table 8.6: **The Plot Sensitivity Utility Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Sensitivity function**	**Sensitivity function**
	Standard reduced frames	Standard reduced frames

8.10 Show Reduced Spectra

This utility offers a wide range of tools to visualize the reduced spectra. It works both on the `BFCR` (`DFCR`) frames and on the combined frames.

It allows to:

- visualize the 2D and 1D spectra;
- visualize the 1D error spectrum;
- visualize the 1D sky spectrum at the target position;
- visualize the extraction profile and the extraction region;

- edit the 1D spectrum;
- check the target redshift;
- fit the emission/absorption lines;
- perform some "simple" statistic (e.g. S/N measurements in a selected region, median, rms).

For more details on its use, see the *Show Reduced Spectra* description in the Cookbook section.

Table 8.7: **The Show Reduced Utility Summary**. In bold mandatory inputs

	MODS	LUCI
INPUTS	**Reduced or combined frame**	**Reduced or combined frame**
	Spectro-photometric standars	Stellar template

8.11 Manage Detections

This task allows to manage the objects detections. Both the *Reduce Observations* and the *Combine Observations* recipe detect objects as described *here*. However, the user may want to modify these detections (and update the corresponding 1D spectra), extract new non-detected objects or delete some detections (and their relative 1D spectra). This task allows the user to do all these operations.

It open a new window and shows the selected reduced or combined frame, and for each slit, it shows the extraction profile and a summary of all the objects that have been detected by the recipes in that slit.

For more details on its use, see the *Manage Detections* description in the Cookbook section.

Table 8.8: **The Manage Detections Utility Summary**. In bold mandatory inputs

	MODS	LUCI
INPUTS	**Reduced or combined frame**	**Reduced or combined frame**

8.12 Plot Extraction Profile

This task shows the extraction profiles of the selected frame (see the *extraction profile* paragraph).

It opens a new window and shows the extraction profile (grey shaded area) of the slit shown at the bottom of the panel. The green solid line is the median value of the profile, the red horizontal line is the (median + rms) and the blue line is the detection threshold. The grey vertical lines indicate the extraction limits of the object shown at the bottom of the panel.

Table 8.9: **The Plot Extraction Profile Utility Summary**. In bold mandatory inputs

	MODS	LUCI
INPUTS	**Reduced or combined frame**	**Reduced or combined frame**

8.13 Show Window Table

It shows the window table of the selected file. This table exists only in reduced frames. It stores the information both on EXR2D structure and on the detection of 1D spectra.

All 2D extracted spectra are stacked up in the EXR2D following the order explained in the *Show Slit Position* Note. Every extracted slit covers the wavelength range from *WLEN START* to *WLEN END* with a *WLEN INC* sampling (see *Grism tables* keywords or `WLEN_START`, `WLEN_END`, and `WLEN_INC` parameter in *Reduce Observations*).

The Window Table relevant columns are (see Fig. 8.4):

- *SLIT*: sequential SIPGI slit number (always starting from 1)
- *SPEC_START*: first row (Y axis) of the slit in *EXR2D* extension
- *SPEC_END*: last row (Y axis) of the slit in *EXR2D* extension
- *OBJ_START*: first row (Y axis) of object spectrum, relative to *SPEC_START*
- *OBJ_END*: last row (Y axis) of object spectrum, relative to *SPEC_START*
- *OBJ_NO*: sequential number of detected objects within slit
- *OBJ_POS*: peak position of object in 2D extracted spectrum, relative to *SPEC_START*

Each WIN Table has at least 1 entry for each slit. Multiple entries for the same slit can exist in case more than one object was detected in the same slit.

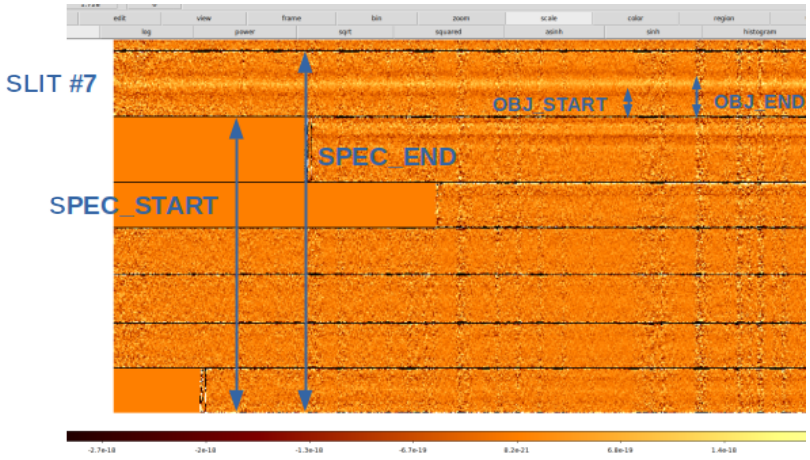

Fig. 8.4: The figure shows the quantity stored in the WIN table. In this example the slit #7 starting from pixel 329 (330 in DS9) and ends at pixel 401 (402 in DS9). One object as been detected from pixel 360 to pixel 371. The peak of the profile object is at pixel 365.899

SLIT	SPEC_START	SPEC_END	OBJ_START	OBJ_END	OBJ_NO	OBJ_POS
1	0	49	19	27	1	22.89
...						
7	**329**	**401**	**30**	**41**	**1**	**35.899**
...						

Table 8.10: **The Show Window Table Utility Summary**. In bold mandatory inputs

	MODS	**LUCI**
INPUTS	**Reduced or combined frame**	**Reduced or combined frame**

CHAPTER

NINE

APPENDIX

9.1 Appendix A: the stellar templates

9.1.1 The Pickles templates

The *Stellar templates* for stars of different spectral type/class SIPGI provides for LUCI flux calibration are listed in the following Table (credits to ESO - Pickles, 1998).

a0i.dat	b3v.dat	f6v.dat	k0iv.dat	m1v.dat	rg0v.dat
a0iii.dat	b57v.dat	8i.dat	k0v.dat	m2.5v.dat	rg5iii.dat
a0iv.dat	b5i.dat	f8iv.dat	k1iii.dat	m2i.dat	rg5v.dat
a0v.dat	b5ii.dat	f8v.dat	k1iv.dat	m2iii.dat	rk0iii.dat
a2i.dat	b5iii.dat	g0i.dat	k2i.dat	m2v.dat	rk0v.dat
a2v.dat	b6iv.dat	g0iii.dat	k2iii.dat	m3ii.dat	rk1iii.dat
a3iiisp.dat	b8i.dat	g0iv.dat	k2v.dat	m3iii.dat	rk2iii.dat
a3v.dat	b8v.dat	g0v.dat	k34ii.dat	m3v.dat	rk3iii.dat
a47iv.dat	b9iii.dat	g2i.dat	k3i.dat	m4iii.dat	rk4iii.dat
a5iii.dat	b9v.dat	g2iv.dat	k3iii.dat	m4v.dat	rk5iii.dat
a5v.dat	f02iv.dat	g2v.dat	k3iv.dat	m5iii.dat	wf5v.dat
a7iii.dat	f0i.dat	g5i.dat	k3v.dat	m5v.dat	wf8v.dat
a7v.dat	f0ii.dat	g5ii.dat	k4i.dat	m6iii.dat	wg0v.dat
b0i.dat	f0iii.dat	g5iii.dat	k4iii.dat	m6v.dat	wg5iii.dat
b0v.dat	f0v.dat	g5iv.dat	k4v.dat	m7iii.dat	wg5v.dat
b12iii.dat	f2ii.dat	g5v.dat	k5iii.dat	m8iii.dat	wg8iii.dat
b1v.dat	f2v.dat	g8iii.dat	k7v.dat	o5v.dat	wk1iii.dat
b1i.dat	f2iii.dat	g8i.dat	k5v.dat	m9iii.dat	wk0iii.dat
b2ii.dat	f5i.dat	g8iv.dat	m0iii.dat	o8iii.dat	wk2iii.dat
b2iv.dat	f5iii.dat	g8v.dat	m0v.dat	o9v.dat	wk3iii.dat
b3i.dat	f5iv.dat	k01ii.dat	m10iii.dat	rf6v.dat	wk4iii.dat
b3iii.dat	f5v.dat	k0iii.dat	m1iii.dat	rf8v.dat	

The sampling is at 5 Angstrom. The templates are **in vacuum**. The conversion from air to vacuum wavelength has been performed by the PyAstronomy library using the Edlen (1953) and Ciddor (1996) prescriptions. The syntax for the filenames is as follows: xxy.dat where xx is the spectral type in lower case letter (e.g. a0, g5), and y the luminosity class in roman lower case letters (i,ii, iii, iv, v) (credits to ESO - Pickles, 1998).

The typical *Stellar template* **must contain two columns and have a constant sampling**: wavelength (Angstrom), fluxes per wavelenght unit (arbitrary units), plus an arbitrary header on top of the file.

9.1.2 The spectro-photometric standards

The *Spectro-photometric standards* SIPGI provides for MODS flux calibration (credits LBTO).

G191-B2B
GD 71
Feige 34
Feige 66
Feige 67
GD 153
Hz 43
Hz 44
BD+332642
BD+284211
Feige 110

The sampling is at 10 Angstrom. The spectra are **in vacuum**.

Note: G191-B2B is an HST primary white dwarf star; GD71 is an HST primary white dwarf standard stars. The NIR extension is based on HST model spectrum; Feige 66 has only relatively low-quality near-IR extension data, and should only be used in the blue or out to about 9200Å GD153 is an HST primary white dwarf star. The NIR extension is based on the HST fluxes shifted by -0.02mag. Hz 43 is an HST primary white dwarf star. The NIR extension is based on the HST model spectrum. A faint binary M-dwarf companion, Hz43B (V=14.3), is located 3-arcsec away, so this star is not recommended during poor seeing. BD+28 4211 has a faint red companion 2.8-arcsec away, and is only recommended for use in the blue channel in good seeing (credits LBTO).

The typical *Spectro-photometric standard* must be composed by three columns: wavelength (Angstrom), magnitude or fluxes, sampling, plus an arbitrary header on top of the file.

9.2 Appendix B: keywords necessary to SIPGI to import and categorize raw data

9.2.1 LUCI

The list of keywords that must be correctly filled in LUCI data for a successful import of raw frames. **Old version** indicates the name of the keyword in old files.

Keyword	Old version
CAMERA	CAMNAME
CD1_1	CROTA1
CD1_2	CDELT1
CD2_1	
CD2_2	
CRPIX1	
CRPIX2	

continues on next page

Table 9.1 – continued from previous page

Keyword	Old version
CRVAL1	
CRVAL2	
CTYPE1	
CTYPE2	
DATE-OBS	
DATATYPE	IMAGETYP
EXPTIME	
FILENAME	
FILTER1	
FILTER2	
FRAMENUM	
GAIN	ELECGAIN
	LBTO LUCI DET EGAIN
GRATNAME	
GRATWLEN	
INSTRUME	
LAMP1	STATLMP1
LAMP2	STATLMP2
LAMP3	STATLMP3
LAMP4	STATLMP4
LAMP5	STATLMP5
LAMP6	STATLMP6
MASKID	MASKNAME
MASKSLOT	
MOSPOS	
NAXIS1	
NAXIS2	
OBJECT	
OBJRA	
OBJDEC	
PI_NAME	
PIXSCALE	
POSANGLE	
TELRA	
TELDEC	

LUCI MOS MASK	
MOS??DEC	Only MOS case
MOS??RA	Only MOS case
MOS??SHA	Only MOS case
MOS??XPO	Only MOS case
MOS??YPO	Only MOS case
MOS??LMM	Only MOS case
MOS??WMM	Only MOS case
TGT??dNAM	Only MOS case

9.2.2 MODS

The list of keywords that must be correctly filled in MODS data for a successful import of raw frames.

CALIB
CALLAMPS
CHANNEL
DATE-OBS
DICHNAME
IMAGETYP
EXPTIME
FILENAME
FILTNAME
FRAMENUM
GRATNAME
INSTRUME
MASKNAME
OBJECT
PI_NAME
PIXSCALE
POSANGLE
NAXIS1
NAXIS2
TELRA
TELDEC

9.3 Appendix C: error propagation on 1D and 2D spectra

9.3.1 Error on preliminary reduced frames

For each raw frame, the Preliminary Reduction recipe estimates the error on the pixel (i,j) of the final pre-reduced frame as:

$$E_{i,j} = \frac{\sqrt{(d_{i,j} \cdot gain + ron^2)}}{gain},$$

where:

- $d_{i,j}$ is the pixel value in LUCI reduction or the pixel value bias subtracted in MODS reduction;
- *gain* is the detector GAIN in [e-/ADU];
- *ron* is the Read Out Noise in [e-].

If in the pre-reduction of LUCI frames a Master Dark is provided, the recipe estimates the error on the input Master Dark as:

$$E_D = rms(D) \cdot \sqrt{gain},$$

where $rms(D)$ is:

$$rms(D) = \sqrt{\frac{\sum_i^N \sum_j^M (D_{i,j} - \bar{D})^2}{MN - 1}},$$

and $D_{i,j}$ is the values of the (i,j)th pixel of the Master Dark, (N,M) the Master Dark dimension in [px] and $\bar{D} = \frac{1}{NM}\sum_i^N \sum_j^M D_{i,j}$. In this case, the final error of the pre-reduced frame dark subtracted is:

$$E_{i,j}^{'} = \sqrt{E_{i,j}^2 + E_D^2}.$$

9.3.2 Error on reduced frames

The error on the reduced frames largely depends on the type of slit, i.e. *distorted* or *un-distorted*, and on the sky subtraction methods.

9.3.2.1 Error for un-distorted slits in reduced frames.

For *un-distorted* slits, the first operation performed by the Reduce Observation recipe is the sky subtraction. Starting from the prereduced frames, the error on each pixel (i,j) of the sky-subtracted image is:

$$E_{SKYSUB,i,j} = \sqrt{E_{i,j}^2 + E_{SKY}^2}$$

where, here and after, $E_{i,j}$ is replaced by $E_{i,j}^{'}$ if the pre-reduced frames are dark subtracted, and E_{SKY} depends on the sky methods. In particular:

- **sky method median**

$$E_{SKY} = \frac{\sqrt{\sum_j E_{i,j}^2}}{N}$$

- **sky method fit/wfit**: in case of weighted fit (wfit), the recipe estimates the weights as:

$$w_j = \frac{1}{E_{i,j}^2},$$

then estimates the weighted fit and computes the chi^2:

$$\chi^2 = \sum_j w_j \left(\bar{d}_{i,j} - \hat{d}_j\right)^2,$$

where $\bar{d}_{i,j}$ is the pixel value of the pre-reduced image, and $\hat{d}_j$ is the value of the weighted fit at the jth pixel.

The final error is:

$$E_{SKY} = \sqrt{\frac{\chi^2}{N - M - 1}},$$

In case of a simple fit w_j = 1.

- **sky method ABBA**:

$$E_{SKY} = E_{i,j}$$

where $E_{i,j}$ is the pixel error of the frame that is *subtracted*.

- **sky method ABBA + median/fit/wfit**: if different sky mehods are combined, the relative errors are added in quadrature.

After the sky subtraction, the recipe rectifies the spectra with the wavelength calibration and extract the 2D spectra. The error on the rectified EXR2D is:

$$E_{EXR2D,i,j} = E_{RESAMPLE,i,j},$$

where $E_{RESAMPLE,i,j}$ is obtained resampling the error of the input pixels (for more details see $r_e(\lambda)$ in *Appendix D*).

The final step is the extraction of 1D spectra. SIPGI offers two possibilities for the spectra extraction: the sum and the Horne extraction. The error associated to the 1D spectra is:

$$E_{EXR1D,i} = \sqrt{\sum_j E^2_{EXR2D,i,j}}$$

where the sum is extended to all the jth pixels of the detected object. In case of a Horne extraction, $E_{EXR1D,i}$ is the error on a weighted sum.

9.3.2.2 Error for distorted slits in reduced frames.

In the reduction of the distorted slit, the first operation is the extraction of the EXR2D. The error associated is:

$$E_{EXR2D,i,j} = E_{RESAMPLE,i,j}.$$

After the extraction of 2D spectra the recipe performs the sky subtraction, and the relative error is:

$$E_{SKYSUB,i,j} = \sqrt{E^2_{EXR2D,i,j} + E^2_{SKY}},$$

where E_{SKY} depends on the sky subtraction method as above:

- **sky method median**

$$E_{SKY} = \frac{\sqrt{\sum_j E^2_{EXR2D,i,j}}}{N}$$

- **sky method fit/wfit**: in case of weighted fit (wfit), the recipe estimates the weights as:

$$w_j = \frac{1}{E^2_{EXR2D,i,j}},$$

then estimates the weighted fit and computes the chi^2:

$$\chi^2 = \sum_j w_j \left(d^*_{i,j} - \hat{d}_j\right)^2,$$

where $d^*_{i,j}$ are the pixel values of the EXR2D image, and $\hat{d}_j$ is the value of the weighted fit at the jth pixel.

The final error is:

$$E_{SKY} = \sqrt{\frac{\chi^2}{N - M - 1}},$$

In case of a simple fit w_j = 1.

- **sky method ABBA**:

$$E_{SKY} = E_{EXR2D,i,j}$$

where $E_{EXR2D,i,j}$ is the pixel error of the EXR2D *subtracted* image.

- **sky method ABBA + median/fit/wfit**: if different sky methods are combined, the relative errors are added in quadrature.

For the distorted slits the DAVIES method is also provided for the sky subtraction but in this version of SIPGI no error propagation is implemented in this case.

The final step is the extraction of the 1D spectra that follow the same scheme of *un-distorted* slits, i.e.:

$$E_{EXR1D,i} = \sum_j E_{SKYSUB,i,j},$$

where the sum is extended to all the jth pixels of the detected object. In case of a Horne extraction, the error is a weighted sum.

9.3.3 Error on combined frames

The error on the (i,j)th pixel of the image obtained combining N frames together is:

$$E_{COMB,i,j} = \frac{\sqrt{\sum_k^N e_{i,j,k}^2}}{N}$$

where $e_{i,j,k}$ is the error of the (i,j)th pixel in the (k)th frame.

9.4 Appendix D: 2D spectra extraction and resampling

The 2D extraction procedure is basically an image warping process and it involves a re-sampling function: the process of transforming the raw spectra from one coordinate system described by the models (*OPT*, *CRV* and *IDS*), to another where the spectra are perfectly rectified and wavelength calibrated.

SIPGI goes along the slices used during the Master Lamp creation (see Fig. 7.5), moving in the coordinates system defined by the 3 models. For each slice and for each wavelength value of the extraction range, it computes the re-sampled flux value $r_f(\lambda)$ and error $r_e(\lambda)$ of the 2D spectrum.

The recipe computes the re-sampling using a filter derived from exponential functions; the library defines a low-pass filter in the Fourier space, using the C^∞ function defined by:

$$H_s(f) = \tanh \frac{s \cdot (f + 0.5) + 1}{2} \tanh \frac{s \cdot (-f + 0.5) + 1}{2}.$$

The parameter s defines the sharpness of the filter and SIPGI uses $s = 5$. The value 0.5 is the cut-off frequency to conform to the Nyquist rate. The actual 1D re-sampling kernel in the image domain, is obtained computing the inverse Fourier transform of $H_s(f)$. This gives us the $h_k(x)$, as shown in Fig. 9.1.

Since the extraction procedure works with 2D images, the 2D re-sampling kernel w is obtained by combining 2 separate 1D kernels along the 2 orthogonal axes x and y

$$w_s(x, y) = h_s(x) \cdot h_s(y).$$

The shape of the 2D re-sampling kernel w_5 is shown in the figure Fig. 9.2.

The 2D extraction process, for each slice and each λ, follows these steps:

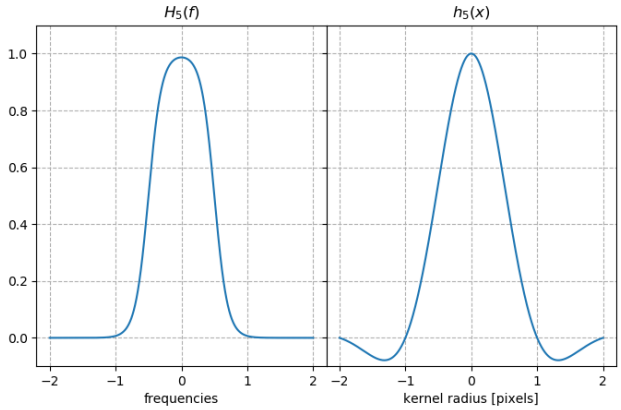

Fig. 9.1: On the left side the $H_5(f)$ spectrum in the Fourier space. On the right side the $h_5(x)$ kernel profile in the image domain, obtained by the inverse fast Fourier transform of $H_5(f)$.

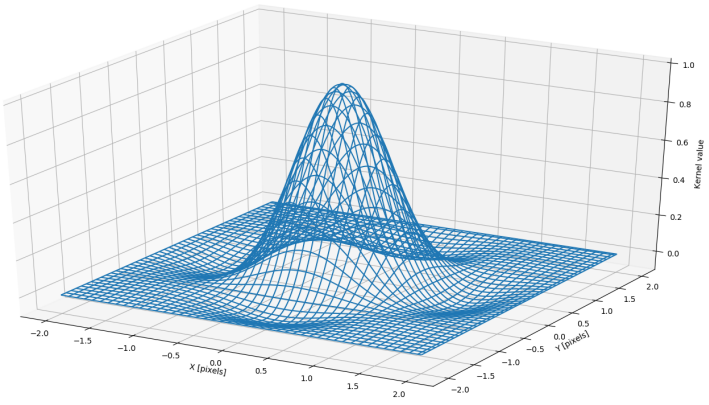

Fig. 9.2: In this figure is shown the shape of the w_5 2D kernel with a radius 2 pixels large.

- the 3 models are used to find the point where the re-sampling kernel must be applied, i.e. the models get the center of the w_5 kernel (red dot in the Fig. 9.3);
- SIPGI collects 16 pixels covered by the the 4x4 pixels box (green pixels in the Fig. 9.3);

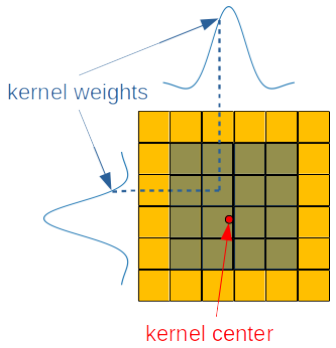

Fig. 9.3: The yellow pixels represent the image pixels; the red dot is the kernel center computed by the models; the blue pixels are the 4x4 kernel box.

- the re-sampled flux value $r_f(\lambda)$ is computed by the formula

$$r_f(\lambda) = \frac{\sum_{x,y} p(x,y) w_5(x,y)}{\sum_{x,y} w_5(x,y)}$$

- The error propagation is compute resampling the errors on pixels in the 4x4 box in the same away

$$r_e(\lambda) = \frac{\sum_{x,y} p_{err}(x,y) w_5(x,y)}{\sum_{x,y} w_5(x,y)}.$$

These steps will be repeated for each re-sampled values and packed into a final spectrum to obtain a wavelength calibrated spectrum corrected by distortions.

9.5 Appendix E: bad pixels correction and cosmic rays cleaning

The bad pixels correction is performed interpolating the good pixels values around the bad pixel. The same algorithm is used to clean the cosmic rays.

Here the recipe step by step, using as example a cluster of 4 bad pixels (A, B, C and D) illustrated in the Fig. 9.4:

- The recipe starts from the bad pixel to correct (B in the example) and it moves along the 4 cardinal directions (North-South, East-West, NorthEast-SouthWest and NorthWest-SouthEast) to find the first good pixels available. The good pixels found in the example are the orange pixels.
- For each direction the recipe should found 2 good pixels: one before (d_b) and one after (d_a) the bad one.
- If both pixels are found the algorithm computes the weighted mean of the two values, using as weights (w_b and w_a) the distances of the d_b and d_a pixels from the bad pixel.

$$m_k = \left(\frac{d_b}{w_b} + \frac{d_a}{w_a}\right) / \left(\frac{1}{w_b} + \frac{1}{w_a}\right), k = 0, 1, 2, 3$$

- On the edges of the frame, the weighted mean along some direction could not be computed because some good pixel candidate could be out of frame. For this reason, we can have less then 4 weighted means. For the example of the pixels B the recipe will found only 3 good m_k values (along the E-W, NE-SW and NW-SE directions) since the pixel after the bad one in the N-S direction falls out of frame.
- The final value of the bad pixel is median of all available m_k values

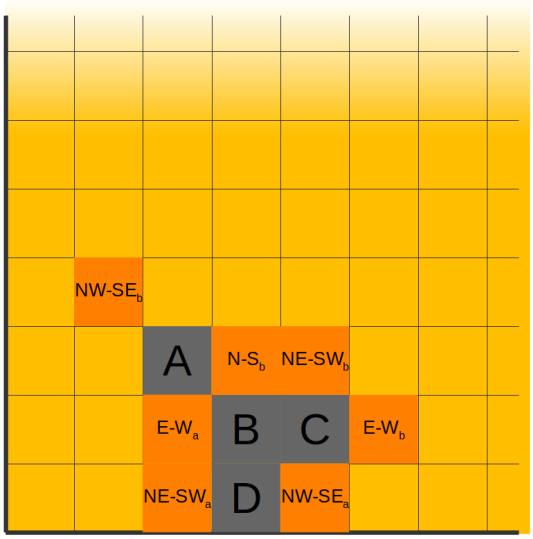

Fig. 9.4: An example of the bad pixel treatment. The yellow pixels represent the image pixels. The gray pixels are a cluster of 4 bad pixels (A, B, C, and D). The orange pixels are the good pixels used by the recipe to correct the central bad pixel B.

CHAPTER

TEN

GLOSSARY

CCD
The CCD Table: FITS table containing info about READOUT NOISE and the list of detected bad pixels

CRV
The Curvature Model: model which describes the spectra curvatures, used to follow the spectra tracing

DataSet
A collection of data of the target or calibration files

DRS
The Data Reduction Software: the reduction core of SIPGI

EXR
The Extraction Table: FITS table containing the solution to extract spectra 2D wavelength calibrated and corrected by distortions

EXR1D
The 1D Extracted Spectra: FITS image containing 1D spectra spectra lambda calibrated, count units

EXR1DFLUX
The 1D Extracted Spectra: FITS image containing 1D spectra spectra lambda calibrated, flux units (erg cm^{-2} s^{-1} $^{-1}$)

EXR2D
The 2D Extracted Spectra: FITS image containing 2D spectra spectra lambda calibrated, count units

EXR2DFLUX
The 2D Extracted Spectra: FITS image containing 2D spectra spectra lambda calibrated, flux units (erg cm^{-2} s^{-1} $^{-1}$)

GRS
The Grism Table: FITS table containing in the HEADER information about extractions limits, spectra re-sampling and lambda calibration refinement

Hippparcos catalog
The FITS file catalog containing Hipparcos stars, with there stellar type and the 2MASS magnitudes (Y, H, K)

IDS
The Inverse Dispersion Solution: model which describes the relation pixel to lambda, used to lambda calibrate spectra

LIN
The Line Catalog: FITS table containing the list of expected emission lines for a given grism

MMS
The MODS Mask files: the ASCII file containing geometrical description of the MODS mask

OPT
The Optical Model: model which describes the instrument optical distortions, used to locate the slits on frame

PAF
The ASCII files containing the first guesses solutions of the 3 SIPGI models: OPT, CRV and IDS

Project
The ensemble of all data needed to achieve a data reduction

Raw sensitivity function
The curve used to obtain the Sensitivity function

Reduction unit
A collection of data of the target acquired with the same instrument configuration

Sensitivity function
The smoothed curve used to convert spectra from ADU to flux units (erg cm^{-2} s^{-1} $^{-1}$)

Spectro-photometric standard
The ASCII file which contains the spectrum of a standard stars

Stellar template
The ASCII file which contains the theoretical spectrum of stars (arbitrary units)

WIN
The Window Table: FITS table containing the description of the slits and information about detected objects

WLEN END
Float FITS keyword in the header of GRS and EXR tables, which defines the ending wavelength () of 2D and 1D extracted spectra

WLEN START
Float FITS keyword in the header of GRS and EXR tables, which defines the starting wavelength () of 2D and 1D extracted spectra

Workspace
The main SIPGI environment

CHAPTER

ELEVEN

REFERENCES

Scodeggio M., Franzetti P., Garilli B., PASP, 117, 1284, 2005;

Garilli B., Paioro L., Scodeggio M., PASP, 124, 1232, 2012;

Pickles A.J., "*A Stellar Spectral Flux Library: 1150-25000 Å*", 1998, PASP, 110, 749. ESA, 1997, The Hipparcos and Tycho Catalogues, ESA SP-1200 Davies Rousellot;

Rousselot et al., A&A 354, 1134-1150, 2000;

Osterbrock et al., PASP, 108, 2270, 1996;

Edlen J., Opt. Soc. Am 43 no. 5, 1953;

Ciddor, Applied Optics 35 no. 9, 1996.

Zacharias et al., AAS…205.4815Z, 2004;

Davies R., MNRAS 375 1099D, 2007;

Horne K., PASP 98 609H, 1986.